

# A Lambda-Calculus Foundation for Universal Probabilistic Programming

Johannes Borgström\*      Ugo Dal Lago†      Andrew D. Gordon‡  
Marcin Szymczak§

May 20, 2016

## Abstract

We develop the operational semantics of an untyped probabilistic  $\lambda$ -calculus with continuous distributions, as a foundation for universal probabilistic programming languages such as Church, Anglican, and Venture. Our first contribution is to adapt the classic operational semantics of  $\lambda$ -calculus to a continuous setting via creating a measure space on terms and defining step-indexed approximations. We prove equivalence of big-step and small-step formulations of this *distribution-based semantics*. To move closer to inference techniques, we also define the *sampling-based semantics* of a term as a function from a trace of random samples to a value. We show that the distribution induced by integrating over all traces equals the distribution-based semantics. Our second contribution is to formalize the implementation technique of trace *Markov chain Monte Carlo* (MCMC) for our calculus and to show its correctness. A key step is defining sufficient conditions for the distribution induced by trace MCMC to converge to the distribution-based semantics. To the best of our knowledge, this is the first rigorous correctness proof for trace MCMC for a higher-order functional language.

---

\*Uppsala University

†University of Bologna & INRIA

‡Microsoft Research and University of Edinburgh

§University of Edinburgh

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Universal Probabilistic Programming in CHURCH . . . . .	3
1.2	Problem 1: Semantics of CHURCH Queries . . . . .	4
1.3	Problem 2: Correctness of Trace MCMC . . . . .	5
1.4	Contributions of the Paper . . . . .	5
1.5	Structure of the Paper . . . . .	5
<b>2</b>	<b>A Foundational Calculus for Church</b>	<b>6</b>
2.1	Syntax of the Calculus . . . . .	6
2.2	Big-step Sampling-based Semantics . . . . .	7
2.3	Encoding CHURCH . . . . .	7
2.4	Example: Geometric Distribution . . . . .	9
2.5	Soft Conditioning . . . . .	9
2.6	Example: Linear Regression . . . . .	10
<b>3</b>	<b>Sampling-Based Operational Semantics</b>	<b>10</b>
3.1	Small-step Sampling-based Semantics . . . . .	11
3.2	Some Measure-Theoretic Preliminaries. . . . .	18
3.3	Measure Space on Program Traces . . . . .	19
3.4	Distribution $\llbracket M \rrbracket_{\mathbb{S}}$ Induced by Sampling-Based Semantics . . . . .	20
<b>4</b>	<b>Distribution-Based Operational Semantics</b>	<b>22</b>
4.1	Subprobability Kernels . . . . .	22
4.2	Approximation Small-Step Semantics . . . . .	22
4.3	Approximation Big-Step Semantics . . . . .	24
4.4	Beyond Approximations . . . . .	25
4.5	Distribution-based and Sampling-based Semantics are Equivalent . . . . .	28
4.6	Motivation for Bounded Scores . . . . .	31
4.7	An Application of the Distribution-Based Semantics . . . . .	31
<b>5</b>	<b>Inference</b>	<b>31</b>
5.1	A Metropolis-Hastings Transition Kernel . . . . .	32
5.2	Correctness of Inference . . . . .	34
5.3	Examples . . . . .	35
<b>6</b>	<b>Related Work</b>	<b>36</b>
<b>7</b>	<b>Conclusions</b>	<b>37</b>
<b>A</b>	<b>Proofs of Measurability</b>	<b>38</b>
A.1	Deterministic reduction as a measurable function . . . . .	42
A.2	Small- step reduction as a measurable function . . . . .	43
A.3	Measurability of $\mathbf{P}$ and $\mathbf{O}$ . . . . .	45
A.4	Measurability of <code>peval</code> . . . . .	47
A.5	Measurability of $q$ and $Q$ . . . . .	51

# 1 Introduction

In computer science, probability theory can be used for models that enable system abstraction, and also as a way to compute in a setting where having access to a source of uniform randomness is essential to achieve correctness, like in randomised computation or cryptography [8]. Domains in which probabilistic models play a key role include robotics [29], linguistics [18], and especially machine learning [24]. The wealth of applications has stimulated the development of concrete and abstract programming languages, that most often are extensions of their deterministic ancestors. Among the many ways probabilistic choice can be captured in programming, the simplest one consists in endowing the language of programs with an operator modelling the sampling from (one or many) distributions. This renders program evaluation a probabilistic process, and under mild assumptions the language becomes universal for probabilistic computation. Particularly fruitful in this sense has been the line of work in the functional paradigm.

In *probabilistic programming*, programs become a way to specify probabilistic models for observed data, on top of which one can later do inference. This has been a source of inspiration for AI researchers, and has recently been gathering interest in the programming language community (see Russell [25] for a survey).

## 1.1 Universal Probabilistic Programming in Church

CHURCH [11] introduced *universal probabilistic programming*, the idea of writing probabilistic models for machine learning in a Turing-complete functional programming language. CHURCH, and its descendants VENTURE [19], ANGLICAN [31], and WEB CHURCH [10] are dialects of SCHEME. Another example of universal probabilistic programming is WEBPPL [9], a probabilistic interpretation of JAVASCRIPT.

A probabilistic query in CHURCH has the form:

```
(query (define x1 e1) ... (define xn en) eq ec)
```

The query denotes the distribution given by the probabilistic expression  $e_q$ , given variables  $x_i$  defined by potentially probabilistic expressions  $e_i$ , conditioned by the boolean predicate  $e_c$ .

Consider a coin with bias  $p$ , that is,  $p$  is the probability of heads. Recall that the *geometric distribution* of the coin is the distribution over the number of flips in a row before it comes up heads. An example of a CHURCH query is as follows: it denotes the geometric distribution of a fair coin, conditioned to be greater than one.

```
(query
  (define flip (lambda (p) (< (rnd) p)))
  (define geometric (lambda (p)
    (if (flip p) 0 (+ 1 (geometric p)))))
  (define n (geometric .5))
  n
  (> n 1))
```

The query defines three variables: (1) `flip` is a function that flips a coin with bias `p`, by calling `(rnd)` to sample a probability from the uniform distribution on the unit interval; (2) `geometric`<sup>1</sup> is a function that samples from the geometric distribution of a coin with bias `p`; and (3) `n` denotes the geometric distribution with bias 0.5. Here are samples from this query:

```
(5 5 5 4 2 2 2 2 3 3 2 2 7 2 2 3 4 2 3)
```

This example is a discrete distribution with unbounded support (any integer greater than one may be sampled with some non-zero probability), defined in terms of a continuous distribution (the uniform distribution on the unit interval). Queries may also define continuous distributions, such as regression parameters.

---

<sup>1</sup>See <http://forestdb.org/models/geometric.html>.

## 1.2 Problem 1: Semantics of Church Queries

The first problem we address in this work is to provide a formal semantics for universal probabilistic programming languages with conditioning. Our example illustrates the common situation in machine learning that models are based on continuous distributions (such as `rnd`) and use conditioning, but previous works on formal semantics for probabilistic  $\lambda$ -calculi do not rigorously treat the combination of these features.

To address the problem we introduce a call-by-value  $\lambda$ -calculus with primitives for random draws from various continuous distributions, and primitives for hard and soft conditioning. We present an encoding of CHURCH into our calculus, and some nontrivial examples of probabilistic models.

We consider two styles of operational semantics for our  $\lambda$ -calculus, in which a term is interpreted in two ways, the first closer to inference techniques, the second more extensional.

**Sampling-Based:** A function from a trace to a value and weight.

**Distribution-Based:** A distribution over terms of our calculus.

To obtain a thorough understanding of the semantics of the calculus, for each of these styles we present two inductive definitions of operational semantics, in small-step and big-step style.

First, we consider the *sampling-based semantics*: the two inductive definitions have the forms shown below, where  $M$  is a closed term,  $s$  is a finite *trace* of random numbers,  $w > 0$  is a *weight*, and  $G$  is a *generalized value*, either a value (constant or  $\lambda$ -abstraction) or the exception `fail` (that represents hard conditioning).

- Figure 4 defines small-step relation  $(M, w, s) \rightarrow (M', w', s')$ .
- Figure 1 defines the big-step relation  $M \Downarrow_w^s G$ .

For example, if  $M$  is a  $\lambda$ -term for our example query and  $M \Downarrow_w^s G$  then there is  $n \geq 0$  such that:

- the trace takes the form  $s = [p_1, \dots, p_{n+1}]$  where  $p_i < 0.5$  if and only if  $i = n + 1$  (the first  $n$  flips tails, the last heads);
- the result takes the form  $G = n$  if  $n > 1$ , and otherwise  $G = \text{fail}$  (failure of the condition predicate leads to `fail`);
- and the weight is  $w = 1$  (density of the uniform distribution).

Theorem 1 shows equivalence: the big-step and small-semantics of a term consume the same traces to produce the same results with the same weights. To interpret these semantics as a probability distribution, we describe a metric space of  $\lambda$ -terms and let  $\mathcal{D}$  range over *distributions*, that is, sub-probability Borel measures on terms of the  $\lambda$ -calculus. We define  $\llbracket M \rrbracket_{\mathcal{S}}$  to be the distribution induced by the sampling-based semantics of  $M$ , by integrating over all traces.

Second, we consider the *distribution-based semantics*, that directly associates distributions with terms, without needing to integrate out traces. The two inductive definitions have the forms shown below, where  $n$  is a step-index:

- Figure 6 defines a family of small-step relations  $M \rightarrow_n \mathcal{D}$ .
- Figure 7 defines a family of big-step relations  $M \Downarrow_n \mathcal{D}$ .

These step-indexed families are approximations to their suprema, distributions written as  $\llbracket M \rrbracket_{\Rightarrow}$  and  $\llbracket M \rrbracket_{\Downarrow}$ . By Theorem 2 we have  $\llbracket M \rrbracket_{\Rightarrow} = \llbracket M \rrbracket_{\Downarrow}$ . The proof of the theorem needs certain properties (Lemmas 33, 36, and 38) that build on compositionality results for subprobability kernels [22] from the measure theory literature. We apply the distribution-based semantics in Section 4.7 to show an equation between two forms of conditioning.

Finally, we reconcile the two rather different styles of semantics: Theorem 3 establishes that  $\llbracket M \rrbracket_{\mathcal{S}} = \llbracket M \rrbracket_{\Rightarrow}$ .

### 1.3 Problem 2: Correctness of Trace MCMC

The second problem we address is implementation correctness. As recent work shows [14, 16], subtle errors in inference algorithms for probabilistic languages are a motivation for correctness proofs for probabilistic inference.

*Markov chain Monte Carlo* (MCMC) is an important class of inference methods, exemplified by the Metropolis-Hastings (MH) algorithm [20, 12], that accumulate samples from a target distribution by exploring a Markov chain. The original work on CHURCH introduced the implementation technique called *trace MCMC* [11]. We consider trace MCMC as an instance of the MH algorithm.

Our final result, Theorem 4, asserts that given a closed term  $M$ , trace MCMC generates a Markov chain of traces,  $s_0, s_1, s_2, \dots$ , whose stationary distribution induces a distribution over values corresponding to the target distribution: the semantics  $\llbracket M \rrbracket$  conditioned on success, that is, that the computation terminates and yields a value (not **fail**). The algorithm is parametric in an MH *proposal kernel*  $Q$ , a function that maps a trace  $s$  of  $M$  to a probability distribution over traces, used to sample the next trace in the Markov chain. We formalize the algorithm rigorously, defining the proposal kernel as a Lebesgue integral of a corresponding density function, that we show to be measurable with respect to the  $\sigma$ -algebra on program traces. We show that the transition kernel  $Q$  and the acceptance ratio of the algorithm satisfy standard criteria: *aperiodicity* and *irreducibility*. Hence, Theorem 4 follows from a classic result of Tierney [30] together with Theorem 3, and the sampling-based semantics, which formalizes the traces used by trace MCMC.

### 1.4 Contributions of the Paper

We make the following original contributions:

1. Definition of an untyped  $\lambda$ -calculus with continuous distributions capable of encoding the core of CHURCH.
2. Development of both sampling-based and distribution-based semantics, shown equivalent (Theorems 1, 2, and 3).
3. First proof of correctness of trace MCMC for a  $\lambda$ -calculus (Theorem 4).

The only previous work on formal semantics of  $\lambda$ -calculi with continuous distributions is recent unpublished work by Staton et al. [27]. Their main contribution is an elegant denotational semantics for a simple-typed  $\lambda$ -calculus with continuous distributions and soft conditioning, but without recursion. They do not consider MCMC inference. Their work does not apply to the recursive functions (such as the geometric distribution in Section 1.1) or data structures (such as lists) typically found in CHURCH programs. For our purpose of conferring formal semantics on CHURCH-family languages, we consider it advantageous to rely on untyped techniques.

The only previous work on correctness of trace MCMC is a recent paper by Hur et al. [14], which was a key influence on our work. A key difference is that our work applies to higher-order languages, and our development appeals directly to the established correctness framework of Tierney [30].

### 1.5 Structure of the Paper

The rest of the paper is organized as follows.

Section 2 defines the syntax of our probabilistic  $\lambda$ -calculus with draws from continuous distributions, and defines a deterministic *sampling-based* operational semantics for our calculus. The semantics is based on the explicit consumption of a program trace  $s$  of random draws and production of an explicit weight  $w$  for each outcome.

Section 3 is concerned with a more in-depth treatment of sampling-based semantics, given in two standard styles: big-step semantics,  $M \Downarrow_w^s G$ , and small-step semantics,  $(M, w, s) \rightarrow (M', w', s')$ , which by

Section 4 defines our step-indexed *distribution-based* operational semantics, in both small-step ( $M \rightarrow_n \mathcal{D}$ ) and big-step ( $M \Downarrow_n \mathcal{D}$ ) styles, which by Theorem 2 are equivalent, and define the meaning  $\llbracket M \rrbracket$  of a term  $M$  to be the supremum of the step-indexed semantics. We end by linking the semantics of this section with those of Section 3: Theorem 3 establishes that  $\llbracket M \rrbracket_{\mathcal{S}} = \llbracket M \rrbracket$ .

Section 5 formalizes trace MCMC for our calculus, in the spirit of Hur et al. [14]. Theorem 4 shows equivalence between the distribution computed by the algorithm and the semantics of the previous sections. Hence, Theorem 4 is the first correctness theorem for trace MCMC for a  $\lambda$ -calculus.

Section 6 describes related work and Section 7 concludes.

Appendix A collects various proofs of measurability.

## 2 A Foundational Calculus for Church

In this section, we describe the syntax of our calculus and equip it with an intuitive semantics relating program outcomes to the sequences of random choices made during evaluation. By translating CHURCH constructs to this calculus, we also show that it can serve as a foundation for Turing complete probabilistic languages.

### 2.1 Syntax of the Calculus

We represent scalar data as real numbers  $c \in \mathbb{R}$ . We use 0 and 1 to represent **false** and **true**, respectively. Let  $\mathcal{I}$  be a countable set of *distribution identifiers* (or simply *distributions*). Metavariables for distributions are  $D, E$ . Each distribution identifier  $D$  has an integer *arity*  $|D| \geq 0$ . Each distribution identifier  $D$  defines a density function  $\text{pdf}_D : \mathbb{R}^{|D|} \times \mathbb{R} \rightarrow [0, \infty)$  of a sub-probability kernel. For example, a draw (**rnd**) from the uniform distribution on the unit interval has density  $\text{pdf}_{\text{rnd}}(c) = 1$  if  $c \in [0, 1]$  and otherwise 0, while a draw (**Gaussian**( $m, v$ )) from the Gaussian distribution with mean  $m$  and variance  $v$  has density  $\text{pdf}_{\text{Gaussian}}(m, v, c) = 1/(e^{\frac{(c-m)^2}{2v}} \sqrt{v2\pi})$  if  $v > 0$  and otherwise 0.

Let  $g$  be a metavariable ranging over a countable set of *function identifiers* each with an integer *arity*  $|g| > 0$  and with an interpretation as a total measurable function  $\sigma_g : \mathbb{R}^{|g|} \rightarrow \mathbb{R}$ . Examples of function identifiers include addition  $+$ , comparison  $>$ , and equality  $=$ ; they are often written in infix notation. We define the *values*  $V$  and *terms*  $M$  as follows, where  $x$  ranges over a denumerable set of variables  $\mathcal{X}$ .

$$\begin{aligned} V ::= & c \mid x \mid \lambda x.M \\ M, N ::= & V \mid M N \mid D(V_1, \dots, V_{|D|}) \mid g(V_1, \dots, V_{|g|}) \\ & \mid \text{if } V \text{ then } M \text{ else } L \mid \text{score}(V) \mid \text{fail} \end{aligned}$$

The term **fail** acts as an exception and models *hard* conditioning. The term **score**( $c$ ) models *soft* conditioning, and is thus parametrized on a positive probability  $c \in (0, 1]$ . As usual, free occurrences of  $x$  inside  $M$  are bound by  $\lambda x.M$ . Terms are taken modulo renaming of bound variables. Substitution of all free occurrences of  $x$  by a value  $V$  in  $M$  is defined as usual, and denoted  $M\{V/x\}$ . This can be easily generalized to  $M\{\vec{V}/\vec{x}\}$ , where  $\vec{x}$  is a sequence of variables and  $\vec{V}$  is a sequence of values (of the same length).  $\Lambda$  denotes the set of all terms, and  $C\Lambda$  the set of *closed* terms. *Erroneous redexes*, ranged over by metavariables like  $T, R$ , are closed terms in one of the following three forms:

- $D(V_1, \dots, V_{|D|})$  where at least one of the  $V_i$  is a  $\lambda$ -abstraction.
- $g(V_1, \dots, V_{|g|})$  where at least one of the  $V_i$  is a  $\lambda$ -abstraction.
- **if**  $V$  **then**  $M$  **else**  $L$ , where  $V$  is neither **true** nor **false**.
- **score**( $V$ ), where  $V \notin (0, 1]$ .

$\frac{G \in \mathcal{GV}}{G \Downarrow_1^\emptyset G} \text{(EVAL VAL)}$	$\frac{w = \text{pdf}_D(\vec{c}, c) \quad w > 0}{D(\vec{c}) \Downarrow_w^{[c]} c} \text{(EVAL RANDOM)}$
$\frac{\text{pdf}_D(\vec{c}, c) = 0}{D(\vec{c}) \Downarrow_0^{[c]} \text{fail}} \text{(EVAL RANDOM FAIL)}$	$\frac{}{g(\vec{c}) \Downarrow_1^\emptyset \sigma_g(\vec{c})} \text{(EVAL PRIM)}$
$\frac{M \Downarrow_{w_1}^{s_1} \lambda x.P \quad N \Downarrow_{w_2}^{s_2} V \quad P[V/x] \Downarrow_{w_3}^{s_3} G}{M \ N \ \Downarrow_{w_1 \cdot w_2 \cdot w_3}^{s_1 @ s_2 @ s_3} G} \text{(EVAL APPL)}$	$\frac{M \Downarrow_w^s \text{fail}}{M \ N \ \Downarrow_w^s \text{fail}} \text{(EVAL APPL RAISE1)}$
$\frac{M \Downarrow_w^s c}{M \ N \ \Downarrow_w^s \text{fail}} \text{(EVAL APPL RAISE2)}$	$\frac{M \Downarrow_{w_1}^{s_1} \lambda x.P \quad N \Downarrow_{w_2}^{s_2} \text{fail}}{M \ N \ \Downarrow_{w_1 \cdot w_2}^{s_1 @ s_2} \text{fail}} \text{(EVAL APPL RAISE3)}$
$\frac{M \Downarrow_w^s G}{\text{if true then } M \text{ else } N \Downarrow_w^s G} \text{(EVAL IF TRUE)}$	$\frac{N \Downarrow_w^s G}{\text{if false then } M \text{ else } N \Downarrow_w^s G} \text{(EVAL IF FALSE)}$
$\frac{c \in (0, 1]}{\text{score}(c) \Downarrow_c^\emptyset \text{true}} \text{(EVAL SCORE)}$	$\frac{T \text{ is an erroneous redex}}{T \Downarrow_1^\emptyset \text{fail}} \text{(EVAL FAIL)}$

Figure 1: Sampling-Based Big Step Semantics

## 2.2 Big-step Sampling-based Semantics

In defining the first semantics of the calculus, we use the classical observation [17] that a probabilistic program can be interpreted as a deterministic program parametrized by the sequence of random draws made during the evaluation. We write  $M \Downarrow_w^s V$  to mean that evaluating  $M$  with the outcomes of random draws as listed in the sequence  $s$  yields the value  $V$ , together with the *weight*  $w$  that expresses how likely this sequence of random draws would be if the program was just evaluated randomly. Because our language has continuous distributions,  $w$  is a probability *density* rather than a probability mass. Similarly,  $M \Downarrow_w^s \text{fail}$  means that evaluation of  $M$  with the random sequence  $s$  fails. In either case, the finite trace  $s$  consists of exactly the random choices made during evaluation, with no unused choices permitted.

Formally, we define *program traces*  $s, t$  to be finite sequences  $[c_1, \dots, c_n]$  of reals of arbitrary length, and *generalized values*  $G, H$  to be elements of the set  $\mathcal{GV} = \mathcal{V} \cup \{\text{fail}\}$ , *i.e.*, generalized values are either values or **fail**.

We let  $M \Downarrow_w^s G$  be the least relation closed under the rules in Figure 1. The (EVAL RANDOM) rule replaces a random draw from a distribution  $D$  parametrized by a vector  $\vec{c}$  with the first (and only) element  $c$  of the trace, presumed to be the outcome of the random draw, and sets the weight to the value of the density of  $D(\vec{c})$  at  $c$ . (EVAL RANDOM FAIL) throws an exception if  $c$  is outside the support of the corresponding distribution. Meanwhile, (EVAL SCORE), applied to  $\text{score}(c)$ , sets the weight to  $c$  and returns a dummy value. The applications of **score** are described in Section 2.5.

All the other rules are standard for a call-by-value lambda-calculus, except that they allow the traces to be split between subcomputations and they multiply the weights yielded by subcomputations to obtain the overall weight.

## 2.3 Encoding Church

We now demonstrate the usefulness and expressive power of the calculus via a translation of CHURCH, an untyped higher-order functional probabilistic language.

$\langle c \rangle_e = c$
$\langle x \rangle_e = x$
$\langle g \ e_1, \dots, e_n \rangle_e =$ $\quad \text{let } x_1 = e_1 \text{ in } \dots \text{let } x_n = e_n \text{ in } g(x_1, \dots, x_n)$ $\quad \text{where } x_1, \dots, x_n \notin \text{fv}(e_1) \cup \dots \cup \text{fv}(e_n)$
$\langle D \ e_1, \dots, e_n \rangle_e =$ $\quad \text{let } x_1 = e_1 \text{ in } \dots \text{let } x_n = e_n \text{ in } D(x_1, \dots, x_n)$ $\quad \text{where } x_1, \dots, x_n \notin \text{fv}(e_1) \cup \dots \cup \text{fv}(e_n)$
$\langle \text{lambda } () \ e \rangle_e = \lambda x. \langle e \rangle_e \quad \text{where } x \notin \text{fv}(e)$
$\langle \text{lambda } x \ e \rangle_e = \lambda x. \langle e \rangle_e$
$\langle \text{lambda } (x_1 \dots x_n) \ e \rangle_e = \lambda x_1. \langle \text{lambda } (x_2 \dots x_n) \ e \rangle_e$
$\langle e_1 \ e_2 \rangle_e = \langle e_1 \rangle_e \ \langle e_2 \rangle_e$
$\langle e_1 \ e_2 \dots e_n \rangle_e = \langle (e_1 \ e_2) \dots e_n \rangle_e$
$\langle \text{if } e_1 \ e_2 \ e_3 \rangle_e = \text{let } x = e_1 \text{ in } (\text{if } x \text{ then } \langle e_2 \rangle_e \text{ else } \langle e_3 \rangle_e)$ $\quad \text{where } x \notin \text{fv}(e_2) \cup \text{fv}(e_3)$
$\langle \text{query } (\text{define } x_1 \ e_1) \dots (\text{define } x_n \ e_n) \ e_{out} \ e_{cond} \rangle =$ $\quad \text{let } x_1 = (\text{fix } x_1. \langle e_1 \rangle_e) \text{ in}$ $\quad \dots$ $\quad \text{let } x_n = (\text{fix } x_n. \langle e_n \rangle_e) \text{ in}$ $\quad \text{let } b = e_{cond} \text{ in}$ $\quad \text{if } b \text{ then } e_{out} \text{ else fail}$

Figure 2: Translation of CHURCH

The syntax of CHURCH's *expressions*, *definitions* and *queries* is described as follows:

$$\begin{aligned}
e ::= & c \mid x \mid (g \ e_1 \dots e_n) \mid (D \ e_1 \dots e_n) \mid (\text{if } e_1 \ e_2 \ e_3) \\
& \mid (\text{lambda } (x_1 \dots x_n) \ e) \mid (e_1 \ e_2 \dots e_n) \\
d ::= & (\text{define } x \ e) \\
q ::= & (\text{query } d_1 \dots d_n \ e \ e_{cond})
\end{aligned}$$

To make the translation more intuitive, it is convenient to add to the target language a let-expression of the form **let**  $x = M$  **in**  $N$ , that can be interpreted as syntactic sugar for  $(\lambda x. N) M$ , and sequencing  $M; N$  that stands for  $\lambda \_ . N \ M$  where  $\_$  as usual stands for a variable that does not appear free in any of the terms under consideration.

The rules for translating CHURCH expressions to the calculus are shown in Figure 2, where  $\text{fv}(e)$  denotes the set of free variables in expression  $e$  and  $\text{fix } x. M$  is the value  $\lambda y. N_{\text{fix}} N_{\text{fix}} (\lambda x. M) y$  where  $N_{\text{fix}}$  is  $\lambda z. \lambda w. w (\lambda y. ((zz)w) y)$ . Observe that  $(\text{fix } x. M) V$  evaluates deterministically to  $M\{(\text{fix } x. M)/x\}V$ . We assume that for each distribution identifier  $D$  of arity  $k$ , there is a deterministic function  $\text{pdf}_D$  of arity  $k + 1$  that calculates the corresponding density at the given point.

In addition to expressions presented here, CHURCH also supports *stochastic memoization* [11] by means of a **mem** function, which, applied to any given function, produces a version of it that always returns the same value when applied to the same arguments. This feature allows for functions of integers to be treated as infinite lazy lists of random values, and is useful in defining some nonparametric models, such as the Dirichlet Process.

It would be straightforward to add support for memoization in our encoding by changing the translation to state-passing style, but we omit this standard extension for the sake of brevity.



## 2.4 Example: Geometric Distribution

To illustrate the sampling-based semantics, recall the geometric distribution example from Section 1. It translates to the following program in the core calculus:

```

let flip = λx.(rnd() < x) in
let geometric =
  (fix geometric.
    λp. (let y = flip p in
        if y then 0 else 1 + (geometric p))) in
let n = geometric 0.5 in
let b = n > 1 in
if b then n else fail

```

Suppose we want to evaluate this program with the random trace  $s = [0.7, 0.8, 0.3]$ . By (EVAL APPL), we can substitute the definitions of *flip* and *geometric* in the remainder of the program, without consuming any elements of the trace nor changing the weight of the sample. Then we need to evaluate *geometric* 0.5.

It can be shown (by repeatedly applying (EVAL APPL)) that for any lambda-abstraction  $\lambda x.M$ ,  $M\{(fix\ x.M)/x\} \ V \Downarrow_w^s G$  implies  $(fix\ x.M) \ V \Downarrow_w^s G$ , which allows us to unfold the recursion. Applying the unfolded definition of *geometric* to the argument 0.5 yields an expression of the form:

```

let y = rnd() < 0.5 in
if y then 0 else 1 + (...)

```

For the first random draw, we have  $\text{rnd}() \Downarrow_1^{[0.7]} 0.7$  by (EVAL RANDOM) (because the density of *rnd* is 1 on the interval  $[0, 1]$ ) and so (EVAL PRIM) gives  $\text{rnd}() < 0.5 \Downarrow_1^{[0.7]} \text{false}$ . After unfolding the recursion two more times, evaluating the subsequent “flips” yields  $\text{rnd}() < 0.5 \Downarrow_1^{[0.8]} \text{false}$  and  $\text{rnd}() < 0.5 \Downarrow_1^{[0.3]} \text{true}$ . By (EVAL IF TRUE), the last if-statement evaluates to 0, terminating the recursion. Combining the results by (EVAL APPL), (EVAL IF FALSE) and (EVAL PRIM), we arrive at  $\text{geometric } 0.5 \Downarrow_1^{[0.7, 0.8, 0.3]} 2$ .

At this point, it is straightforward to see that the condition in the final if-statement is satisfied, and hence the program reduces with the given trace to 2 with weight 1.

The weight yielded by every trace in this program that satisfies the final observation is 1. This may seem counter-intuitive, because clearly not all traces have the same probability. This illustrates the subtle difference between probability and probability density. In fact, in this and similar programs, the probability of a given outcome is “hidden” inside an integral over the space of traces, described in Section 3.

## 2.5 Soft Conditioning

The geometric distribution example in Section 2.4 uses so-called *hard conditioning*: that is, program execution fails and the value of  $n$  is discarded whenever the Boolean predicate  $n > 1$  is not satisfied. The same (normalized) distribution on successful runs would be obtained by re-evaluating the entire program from the beginning whenever the Boolean predicate fails, as in

```

let f = fix f λ_.let...in
  if b then n else f 0 in
f 0

```

This corresponds to a basic inference algorithm known as *rejection sampling*. Still, in many machine learning applications we want to use a different kind of conditioning that models noisy data. For instance, if  $c$  is the known output of a sensor that shows an approximate value of some unknown quantity  $x$  that is computed by the program, we want to assign higher probabilities to values of  $x$  that are closer to  $c$ . This is sometimes referred to as *soft conditioning*, and allows the

use of more efficient gradient-based methods of inference (e.g., Homan and Gelman [13]). One simple form of soft conditioning is to flip a biased coin with a success probability based on  $|x - c|$ , for instance

*condition' x c y* := if *flip*( $\exp(-(m - c)^2)$ ) then *y* else fail.

Then *condition' x c M* has the effect of continuing as *M* with probability  $\exp(-(m - c)^2)$ , and otherwise terminating execution. In the context of a sampling-based semantics, it has the effect of multiplying the weight of any successful trace by  $\exp(-(m - c)^2)$ , and adding a true sample  $p \in (0.5, 1]$  to the trace. This dummy sample, a form of *nuisance parameter*, will exist in all successful traces, and a more efficient implementation of the *condition'* construct would instead avoid adding the sample to the trace, and avoid terminating the execution of the program in all cases. This can be achieved using the primitive **score**, as follows.

*condition x c M* := **score**( $\exp(-(m - c)^2)$ ); *M*

## 2.6 Example: Linear Regression

For an example of soft conditioning, consider the ubiquitous linear regression model  $y = m \cdot x + b + \text{noise}$ , where  $x$  is often a known feature and  $y$  an observable outcome variable. We can model the noise as drawn from a Gaussian distribution with mean 0 and variance 1/2 by letting the success probability be given by the function **squash** below.

The following query<sup>2</sup> predicts the  $y$ -coordinate for  $x = 4$ , given observations of four points: (0, 0), (1, 1), (2, 4), and (3, 6). (We use the abbreviation (**define** ( $f\ x_1 \dots x_n$ )  $e$ ) for (**define**  $f$  (**lambda** ( $x_1 \dots x_n$ )  $e$ ), and use **and** for multiadic conjunction.)

```
(query
  (define (sqr x) (* x x))
  (define (squash x y) (exp(- (sqr(- x y)))))
  (define (flip p) (< (rnd) p))
  (define (softeq x y) (flip (squash x y)))

  (define m (gaussian 0 2))
  (define b (gaussian 0 2))
  (define (f x) (+ (* m x) b))

  (f 4)    ;; predict y for x=4

  (and (softeq (f 0) 0) (softeq (f 1) 1)
       (softeq (f 2) 4) (softeq (f 3) 6))
```

The model described above puts independent Gaussian priors on  $m$  and  $b$ . The condition of the query states that all observed  $y$ s are (soft) equal to  $k \cdot x + m$ . Assuming that **softeq** is used only to define conditioning (i.e., positively), we can avoid the nuisance parameter that arises from each **flip** by redefining **softeq** as follows (given a **score** primitive in CHURCH, mapped to **score**(-) in our  $\lambda$ -calculus):

```
(define (softeq x y) (score (squash x y)))
```

## 3 Sampling-Based Operational Semantics

In this section, we further investigate sampling-based semantics for our calculus. First, we will introduce *small*-step sampling-based semantics and prove it equivalent to its big-step sibling as

<sup>2</sup>Cf. <http://forestdb.org/models/linear-regression.html>.

$$\begin{array}{c}
E[g(\vec{c})] \xrightarrow{\text{det}} E[\sigma_g(\vec{c})] \\
E[(\lambda x.M) V] \xrightarrow{\text{det}} E[M\{V/x\}] \\
E[c V] \xrightarrow{\text{det}} E[\text{fail}] \\
E[\text{if } 1 \text{ then } M_2 \text{ else } M_3] \xrightarrow{\text{det}} E[M_2] \\
E[\text{if } 0 \text{ then } M_2 \text{ else } M_3] \xrightarrow{\text{det}} E[M_3] \\
E[T] \xrightarrow{\text{det}} E[\text{fail}] \\
E[\text{fail}] \xrightarrow{\text{det}} \text{fail} \quad \text{if } E \text{ is not } [\cdot]
\end{array}$$

Figure 3: Deterministic Reduction.

introduced in Section 2.2. Then, we will associate to any closed term  $M$  two sub-probability distributions: one on the set of random traces, and the other on the set of return values. This requires some measure theory, recalled in Section 3.2.

### 3.1 Small-step Sampling-based Semantics

We define small-step call-by-value evaluation. The set of all *closed values* is  $\mathcal{V}$ , and we write  $\mathcal{V}_\lambda$  for  $\mathcal{V} \setminus \mathbb{R}$ . *Evaluation contexts* are defined as follows:

$$E ::= [\cdot] \mid EM \mid (\lambda x.M)E$$

We let  $\mathcal{C}$  be the set of all closed evaluation contexts, i.e., where every occurrence of a variable  $x$  is as a subterm of  $\lambda x.M$ .

The term obtained by replacing the only occurrence of  $[\cdot]$  in  $E$  by  $M$  is indicated as  $E[M]$ . *Redexes* are generated by the following grammar:

$$\begin{array}{l}
R ::= (\lambda x.M)V \mid cM \mid D(\vec{c}) \mid g(\vec{c}) \mid \text{score}(c) \\
\mid \text{fail} \mid \text{if true then } M \text{ else } N \\
\mid \text{if false then } M \text{ else } N \mid T
\end{array}$$

*Reducible terms* are those closed terms  $M$  that can be written as  $E[R]$ .

**Lemma 1** *For every closed term  $M$ , either  $M$  is a generalized value or there are unique  $E, R$  such that  $M = E[R]$ . Moreover, if  $M$  is not a generalized value and  $R = \text{fail}$ , then  $E$  is proper, that is,  $E \neq [\cdot]$ .*

PROOF. This is an easy induction on the structure of  $M$ . □

*Deterministic reduction* is the relation  $\xrightarrow{\text{det}}$  on closed terms defined in Figure 3.

**Lemma 2** *If  $M \xrightarrow{\text{det}} M'$  and  $M \xrightarrow{\text{det}} M''$  then  $M' = M''$ .*

PROOF. Since  $M \xrightarrow{\text{det}} M'$  implies that  $M$  is not a generalized value, Lemma 1 states that  $M = E[R]$  for some unique  $E, R$ . If  $R = \text{fail}$ , then  $E$  is proper and  $E[R]$  can only reduce to  $\text{fail}$ . Otherwise, it follows immediately by inspection of the reduction rules that  $E[R] \xrightarrow{\text{det}} E[N]$  for some  $N$  that is uniquely determined by the redex  $R$ . □

$$\boxed{
\begin{array}{c}
\frac{w' = \text{pdf}_D(\vec{c}, c) \quad w' > 0}{(E[D(\vec{c})], w, c :: s) \rightarrow (E[c], ww', s)} \text{(RED RANDOM)} \\
\\
\frac{\text{pdf}_D(\vec{c}, c) = 0}{(E[D(\vec{c})], w, c :: s) \rightarrow (E[\text{fail}], 0, s)} \text{(RED RANDOM FAIL)} \\
\\
\frac{M \rightarrow N}{(M, w, s) \rightarrow (N, w, s)} \text{(RED PURE)} \\
\\
\frac{c \in (0, 1]}{(E[\text{score}(c)], w, s) \rightarrow (E[\text{true}], cw, s)} \text{(RED SCORE)}
\end{array}
}$$

Figure 4: Small-step sampling-based operational semantics

Let us define composition of contexts  $E \circ E'$  inductively as:

$$\begin{aligned}
[\ ] \circ E' &\triangleq E' \\
(E\ M) \circ E' &\triangleq (E \circ E')\ M \\
((\lambda x.M)\ E) \circ E' &\triangleq (\lambda x.M)\ (E \circ E')
\end{aligned}$$

**Lemma 3**  $(E \circ E')[M] = E[E'[M]]$ .

PROOF. By induction on the structure of  $E$ . □

**Lemma 4** If  $E[R] \xrightarrow{\text{det}} E[N]$ , then  $R \xrightarrow{\text{det}} N$ .

PROOF. By case analysis on the deterministic reduction rules.

**Lemma 5** For any  $E$  and  $M$  such that  $M \neq E'[\text{fail}]$ , if  $M \xrightarrow{\text{det}} M'$  then  $E[M] \xrightarrow{\text{det}} E[M']$ .

PROOF. Standard, using Lemmas 1 and 2.

Since  $M \xrightarrow{\text{det}} M'$ ,  $M$  is not a generalized value. By Lemma 1,  $M = E'[R]$  for some  $E', R$ .

By assumption,  $R \neq \text{fail}$ , so by inspection of the reduction rules  $E'[R] \xrightarrow{\text{det}} E'[N]$  for some  $N$ . By Lemma 2,  $E'[N] = M'$ . By Lemma 3,  $E[M] = (E \circ E')[R]$  and  $E[M'] = (E \circ E')[N]$ .

Since Lemma 4 gives  $R \xrightarrow{\text{det}} N$ , by case analysis on the derivation of  $R \xrightarrow{\text{det}} N$  we can show that  $(E \circ E')[R] \xrightarrow{\text{det}} (E \circ E')[N]$ , which implies  $E[M] \xrightarrow{\text{det}} E[M']$ . □

**Lemma 6** If  $(M, w, s) \rightarrow (M', w', s')$  and  $(M, w, s) \rightarrow (M'', w'', s'')$ , then  $M' = M''$ ,  $w' = w''$  and  $s'' = s'$ .

PROOF. By case analysis. Since there is no rule that reduces generalized values,  $(M, w, s) \rightarrow (M', w', s')$  implies that  $M \notin \mathcal{GV}$ , so by Lemma 1,  $M = E[R]$  for some unique  $E, R$ .

- If  $(M, w, s) \rightarrow (M', w', s')$  was derived with (RED PURE), then  $M = E[R]$ , where  $R \neq D(\underline{c})$  and  $R \neq \text{score}(c)$ , which implies that  $(M, w, s) \rightarrow (M'', w'', s'')$  must also have been derived with (RED PURE). Hence, we have  $w'' = w' = w$ ,  $s'' = s' = s$ ,  $M \xrightarrow{\text{det}} M'$  and  $M \xrightarrow{\text{det}} M''$ . By Lemma 2,  $M'' = M'$ , as required.
- If  $(M, w, s) \rightarrow (M', w', s')$  was derived with (RED RANDOM), then  $M = E[D(\vec{c})]$ ,  $s = c :: s^*$  and  $\text{pdf}_D(\vec{c}, c) > 0$ . Hence,  $(M, w, s) \rightarrow (M'', w'', s'')$  must also have been derived with (RED RANDOM), and so  $M'' = M' = E[c]$ ,  $s'' = s' = s^*$  and  $w'' = w' = w \text{pdf}_D(\vec{c}, c)$ , as required. The (RED RANDOM FAIL) case is analogous.

- If  $(M, w, s) \rightarrow (M', w', s')$  was derived with (RED SCORE), then  $M = E[\text{score}(c)]$  and  $c \in (0, 1]$ , so  $(M, w, s) \rightarrow (M'', w'', s'')$  must also have been derived with (RED SCORE). Hence  $M'' = M' = E[\text{true}]$ ,  $w'' = w' = c \cdot w$  and  $s'' = s' = s$ .

□

Rules of small-step reduction are given in Figure 4. We let *multi-step reduction* be the inductively defined relation  $(M, w, s) \Rightarrow (M', w', s')$  if and only if  $(M, w, s) = (M', w', s')$  or  $(M, w, s) \rightarrow (M'', w'', s'') \Rightarrow (M', w', s')$  for some  $M'', w'', s''$ . As can be easily verified, the multi-step reduction of a term is deterministic once the underlying trace and weight are kept fixed:

**Lemma 7** *If both  $(M, w, s) \Rightarrow (G', w', s')$  and  $(M, w, s) \Rightarrow (G'', w'', s'')$ , then  $G' = G''$ ,  $w' = w''$  and  $s' = s''$ .*

PROOF. By induction on the derivation of  $(M, w, s) \Rightarrow (G', w', s')$ , with appeal to Lemma 6.

- Base case:  $(M, w, s) = (G', w', s')$ . Generalized values do not reduce, so  $G'' = G' = G$ ,  $w'' = w' = w$  and  $s'' = s' = s$ .
- Induction step:  $(M, w, s) \rightarrow (\hat{M}, \hat{w}, \hat{s}) \Rightarrow (G', w', s')$ . Since  $M \neq G''$ , we also have  $(M, w, s) \rightarrow (M^*, w^*, s^*) \Rightarrow (G'', w'', s'')$ .  
By Lemma 6,  $(M^*, w^*, s^*) = (\hat{M}, \hat{w}, \hat{s})$ , and so by induction hypothesis,  $(G'', w'', s'') = (G', w', s')$ , as required.

□

**Lemma 8** *For any  $E$  and  $M$  such that  $M \neq E'[\text{fail}]$ , if  $(M, w, s) \rightarrow (M', w', s')$  then  $(E[M], w, s) \rightarrow (E[M'], w', s')$*

PROOF. By inversion of  $\rightarrow$ , using Lemma 5.

- If  $(M, w, s) \rightarrow (M', w', s')$  was derived with (RED PURE), then  $M \xrightarrow{\text{det}} M'$ , so by Lemma 5,  $E[M] \xrightarrow{\text{det}} E[M']$ , and by (RED PURE),  $(E[M], w, s) \rightarrow (E[M'], w', s')$ .
- If  $(M, w, s) \rightarrow (M', w', s')$  was derived with (RED RANDOM), then  $M = E'[\text{D}(\vec{c})]$ ,  $M' = E'[c]$ ,  $s = c :: s'$  and  $w' = w \text{pdf}_{\text{D}}(\vec{c}, c)$ , where  $\text{pdf}_{\text{D}}(\vec{c}, c) > 0$ . By (RED RANDOM) and Lemma 3, we can derive  $(E[M], w, s) \rightarrow (E[M'], w', s')$ . Cases (RED RANDOM FAIL) and (RED SCORE) are analogous.

□

**Lemma 9** *If  $(E[R], w, s) \rightarrow (E[N], w', s')$  then  $(R, w, s) \rightarrow (N, w', s')$*

PROOF. By case analysis.

- If  $(E[R], w, s) \rightarrow (E[N], w', s')$  was derived with (RED PURE), then  $E[R] \xrightarrow{\text{det}} E[N]$ , so by Lemma 5,  $R \xrightarrow{\text{det}} N$ , which implies  $(R, w, s) \rightarrow (N, w', s')$ .
- If  $(E[R], w, s) \rightarrow (E[N], w', s')$  was derived with (RED RANDOM), then  $R = \text{D}(\vec{c})$ ,  $N = c$ ,  $s = c :: s'$  and  $w' = w \text{pdf}_{\text{D}}(\vec{c}, c)$ , where  $\text{pdf}_{\text{D}}(\vec{c}, c) > 0$ .  
Hence, with (RED RANDOM), we can derive  $(\text{D}(\vec{c}), w, s) \rightarrow (c, w', s')$   
Cases (RED RANDOM FAIL) and (RED SCORE) are analogous.

Reduction can take place in any evaluation context, provided the result is not a failure. Moreover, multi-step reduction is a transitive relation. This is captured by the following lemmas.

**Lemma 10** *For any  $E$ , if  $(M, w, s) \Rightarrow (M', w', s')$  and  $M' \neq \text{fail}$ , then we have  $(E[M], w, s) \Rightarrow (E[M'], w', s')$ .*

PROOF. By induction on the number of steps in the derivation of  $(M, w, s) \Rightarrow (M', w', s')$ , with appeal to Lemma 8.

Since  $M' \neq \text{fail}$ , no expression in the derivation chain (other than the last one) can be of the form  $E'[\text{fail}]$ . □

**Lemma 11** *For any  $E$ , if  $(M, w, s) \Rightarrow (\text{fail}, w', s')$  then  $(E[M], w, s) \Rightarrow (\text{fail}, w', s')$ .*

PROOF. By induction on the number of steps in the derivation, using Lemmas 8 and 10. If  $E = []$ , the result holds trivially, so let us assume  $E \neq []$ . If  $(M, w, s) \Rightarrow (\text{fail}, w', s')$  was derived in 0 steps, then  $M = \text{fail}$ ,  $w' = w$  and  $s' = s$ , so by (RED PURE),  $(E[\text{fail}], w, s) \rightarrow (\text{fail}, w, s)$ , as required.

If  $(M, w, s) \Rightarrow (\text{fail}, w', s')$  was derived in 1 or more steps, then:

- If  $M = E'[\text{fail}]$  and  $E' \neq []$ , then  $((E \circ E')[\text{fail}], w, s) \rightarrow (\text{fail}, w', s')$  by (RED PURE).
- Otherwise, there exist  $\hat{M}$ ,  $\hat{w}$ ,  $\hat{s}$  such that  $(M, w, s) \rightarrow (\hat{M}, \hat{w}, \hat{s}) \Rightarrow (\text{fail}, w', s')$ , where  $M \notin \mathcal{GV}$ . By induction hypothesis,  $(E[\hat{M}], \hat{w}, \hat{s}) \Rightarrow (\text{fail}, w', s')$  for any  $E$ , and by Lemma 8,  $(E[M], w, s) \rightarrow (E[\hat{M}], \hat{w}, \hat{s})$ .

□

**Lemma 12** *If  $(M, w, s) \Rightarrow (M', w', s')$  and  $w \geq 0$ , then  $w' \geq 0$ .*

PROOF. By induction on the number of steps in the derivation.

- If  $(M, w, s) \Rightarrow (M', w', s')$  was derived in 0 steps, then  $w' = w$ , so  $w' \geq 0$ .
- If  $(M, w, s) \Rightarrow (M', w', s')$  was derived in 1 or more steps, then  $(M, w, s) \rightarrow (M^*, w^*, s^*) \Rightarrow (M', w', s')$ .  
 If  $(M, w, s) \rightarrow (M^*, w^*, s^*)$  was derived with (RED PURE), then  $w^* = w \geq 0$ .  
 If  $(M, w, s) \rightarrow (M^*, w^*, s^*)$  was derived with (RED RANDOM), then  $w^* = w \cdot w''$  for some  $w'' > 0$ , so  $w^* \geq 0$ .  
 If  $(M, w, s) \rightarrow (M^*, w^*, s^*)$  was derived with (RED SCORE), then  $w^* = w \cdot c$  for some  $c > 0$ , so  $w^* \geq 0$ .  
 If  $(M, w, s) \rightarrow (M^*, w^*, s^*)$  was derived with (RED RANDOM FAIL), then  $w^* = 0$ .  
 In either case,  $w^* \geq 0$ , so by induction hypothesis,  $w' \geq 0$ .

□

**Lemma 13** *If  $(M, w, s) \rightarrow (M', w', s')$  was not derived with (RED RANDOM FAIL) and  $w > 0$ , then  $w' > 0$ .*

PROOF. By inspection (similar to the inductive step in the proof of Lemma 12).

**Lemma 14** *If  $(M, w, s) \rightarrow (M', w', s')$ , then for any  $w^* \geq 0$ ,  $(M, ww^*, s) \rightarrow (M', w'w^*, s')$*

PROOF. By case analysis.

□

**Lemma 15** *If  $(M, w, s) \rightarrow (M', w', s')$ , then for any  $s^*$ ,  $(M, w, s@s^*) \rightarrow (M', w', s'@s^*)$*

PROOF. By case analysis.

□

**Lemma 16** *If  $(M, w, s) \rightarrow (M', w', s')$ , then there is  $s^*$  such that  $s = s^*@s'$  and  $(M, w, s^*) \rightarrow (M', w', [])$*

PROOF. By case analysis.

□

**Lemma 17** *If  $(M, w, s) \rightarrow^k (M', w', s')$ , then for any  $w^* \geq 0$ ,  $(M, ww^*, s) \rightarrow^k (M', w'w^*, s')$*

PROOF. By induction on  $k$ , with appeal to Lemma 14.

□

**Lemma 18** *If  $(M, w, s) \rightarrow^k (M', w', s^*)$ , then for any  $s'$ ,  $(M, w, s@s') \rightarrow^k (M', w', s^*@s')$*

PROOF. By induction on  $k$ , with appeal to Lemma 15.

**Lemma 19** *If both  $(M, 1, s) \Rightarrow (M', w', [])$  and  $(M', 1, s') \Rightarrow (M'', w'', [])$ , then  $(M, 1, s@s') \Rightarrow (M'', w''w'', [])$ .*

PROOF. By Lemma 18,  $(M, 1, s@s') \Rightarrow (M', w', s')$  and by lemma 12,  $w' \geq 0$ . Hence, by Lemma 17,  $(M', w', s') \Rightarrow (M'', w'w'', \square)$ , which gives  $(M, 1, s@s') \Rightarrow (M'', w'w'', \square)$ .  $\square$

**Lemma 20** *For any  $E$ ,  $E[\text{fail}] \Downarrow_1^\square \text{fail}$ .*

PROOF. By induction on the structure of  $E$ .

- Base case:  $E = []$ , the result follows by (EVAL VAL).
- Induction step:
  - Case  $E = (\lambda x.L) E'$ : By induction hypothesis,  $E'[\text{fail}] \Downarrow_1^\square \text{fail}$ , and by (EVAL APPL RAISE2),  $(\lambda x.L) E'[\text{fail}] \Downarrow_1^\square \text{fail}$ , as required.
  - Case  $E = E' L$ : By induction hypothesis,  $E'[\text{fail}] \Downarrow_1^\square \text{fail}$ , so by (EVAL APPL RAISE1), we get  $E'[\text{fail}] L \Downarrow_1^\square \text{fail}$ .

$\square$

**Lemma 21** *For any  $E$ , if  $\text{pdf}_D(\vec{c}, c) = 0$ , then  $E[D(\vec{c})] \Downarrow_0^{[c]} \text{fail}$ .*

PROOF. By induction on the structure of  $E$ .

- Base case:  $E = []$ , the result follows by (EVAL RANDOM FAIL).
- Induction step:
  - Case  $E = (\lambda x.L) E'$ : By induction hypothesis,  $E'[D(\vec{c})] \Downarrow_0^{[c]} \text{fail}$ , and by (EVAL APPL RAISE2),  $(\lambda x.L) E'[D(\vec{c})] \Downarrow_0^\square \text{fail}$ , as required.
  - Case  $E = E' L$ : By induction hypothesis,  $E'[D(\vec{c})] \Downarrow_0^{[c]} \text{fail}$ , so by (EVAL APPL RAISE1), we get  $E'[D(\vec{c})] L \Downarrow_0^\square \text{fail}$ .

$\square$

The following directly relates small-and-big-step semantics, saying that the latter is invariant on the former:

**Lemma 22** *If  $(M, 1, s) \rightarrow (M', w, \square)$  and  $M' \Downarrow_w^{s'} G$ , then  $M \Downarrow_{w.w}^{s@s'} G$ .*

PROOF. By induction on the structure of  $M$ .

If  $M = E[\text{fail}]$  for some  $E \neq []$ , the result follows immediately by Lemma 20. Now, let us assume that  $M \neq E[\text{fail}]$ .

- Base case:  $M = R$ :
  - If  $M = g(\vec{c})$  or  $M = c V$  or  $M = T$ , then  $M$  reduces to a generalized value in 1 step, so the result holds trivially (by one of the evaluation rules).
  - Case  $M = \text{if true then } M_2 \text{ else } M_3$ : We have  $(\text{if true then } M_2 \text{ else } M_3, 1, \square) \rightarrow (M_2, 1, \square)$ . By assumption,  $M_2 \Downarrow_w^{s'} G$ . Thus, the desired result holds by (EVAL IF TRUE).
  - Case  $M = \text{if false then } M_2 \text{ else } M_3$ : analogous to the previous case.
  - Case  $M = (\lambda x.N_1) V$ : We have  $((\lambda x.N_1) V, 1, \square) \rightarrow (N_1\{V/x\}, 1, \square)$ . Since  $(\lambda x.N_1)$  and  $V$  are already values and  $N_1\{V/x\} \Downarrow_w^{s'} G$  by assumption, (EVAL APPL) yields  $(\lambda x.N_1) V \Downarrow_w^{s'} G$ .
  - Case  $M = D(\vec{c})$ :  $(M, 1, s) \rightarrow (M', w, \square)$  must have been derived with (RED RANDOM) or (RED RANDOM FAIL). In the former case,  $s = [c]$ ,  $M' = c$ , and  $w = \text{pdf}_D(\vec{c}, c)$ , where  $c > 0$ . The second assumption then takes the form  $c \Downarrow_1^\square c$ , so the required result follows from (EVAL RANDOM). The (RED RANDOM FAIL) case is similar, with the result following from (EVAL RANDOM FAIL).
  - Case  $M = \text{score}(c)$ ,  $c \in (0, 1]$ :  $(M, 1, s) \rightarrow (M', w, \square)$  must have been derived with (RED SCORE).so  $M' = \text{true}$ ,  $w = c$  and  $s = \square$ . Thus, the result then follows from (EVAL SCORE).
- Induction step:  $M = E[R]$ ,  $E \neq []$ ,  $R \neq \text{fail}$ :
  - Case  $E = (\lambda x.L) E'$ :  $M = (\lambda x.L) E'[R]$ . We have  $((\lambda x.L) E'[R], 1, s) \rightarrow ((\lambda x.L) E'[N], w, \square)$  for some  $N$ , so by lemmas 8 and 9,  $(E'[R], 1, s) \rightarrow (E'[N], w, \square)$ . By assumption,  $(\lambda x.L) E'[N] \Downarrow_w^{s'} G$ .



- If  $(\lambda x.L) E'[N] \Downarrow_w^{s'} G$  was derived with (EVAL APPL), then  $E'[N] \Downarrow_{w_1}^{s_1} V$  and  $(\lambda x.L) V \Downarrow_{w_2}^{s_2} G$ , where  $w' = w_1 w_2$  and  $s' = s_1 @ s_2$ . By induction hypothesis,  $E'[R] \Downarrow_{w_1}^{s @ s_1} V$ , so (EVAL APPL) gives  $(\lambda x.L) E'[R] \Downarrow_{w w_1}^{s @ s'} G$ , as required.
- If  $(\lambda x.L) E'[N] \Downarrow_w^{s'} G$  was derived with (EVAL APPL RAISE3), then  $G = \text{fail}$  and  $E'[N] \Downarrow_w^{s'} \text{fail}$ . By induction hypothesis,  $E'[R] \Downarrow_{w w'}^{s @ s'} \text{fail}$ , so by (EVAL APPL RAISE3),  $(\lambda x.L) E'[R] \Downarrow_{w w'}^{s @ s'} \text{fail}$ .
- Case  $E = E' L$ :  $M = E'[M^*] L$ :  
We have  $(E'[R] L, 1, s) \rightarrow (E'[N] L, w, [])$  for some  $N$ , so by lemmas 8 and 9,  $(E'[R], 1, s) \rightarrow (E'[N], w, [])$ . By assumption,  $E'[N] L \Downarrow_w^{s'} G$ .
  - If  $E'[N] L \Downarrow_w^{s'} G$  was derived with (EVAL APPL), then  $E'[N] \Downarrow_{s_1}^{w_1} (\lambda x.N')$ ,  $L \Downarrow_{s_2}^{w_2} V$  and  $N'[V/x] \Downarrow_{s_3}^{w_3} G$ , where  $w' = w_1 w_2 w_3$  and  $s' = s_1 @ s_2 @ s_3$ . By induction hypothesis,  $E'[R] \Downarrow_{w w_1}^{s @ s_1} (\lambda x.N')$ , so (EVAL APPL) gives  $E'[R] L \Downarrow_{w w'}^{s @ s'} G$ , as required.
  - If  $E'[N] L \Downarrow_w^{s'} G$  was derived with (EVAL APPL RAISE1), then  $G = \text{fail}$  and  $E'[N] \Downarrow_w^{s'} \text{fail}$ . By induction hypothesis,  $E'[R] \Downarrow_{w w'}^{s @ s'} \text{fail}$ , so by (EVAL APPL RAISE1),  $E'[R] L \Downarrow_{w w'}^{s @ s'} \text{fail}$ .
  - If  $E'[N] L \Downarrow_w^{s'} G$  was derived with (EVAL APPL RAISE3), then  $E'[N] \Downarrow_{w_1}^{s_1} (\lambda x.N')$  and  $L \Downarrow_{w_2}^{s_2} \text{fail}$ , where  $w' = w_1 w_2$  and  $s' = s_1 @ s_2$ . By induction hypothesis,  $E'[R] \Downarrow_{w w_1}^{s @ s_1} (\lambda x.N')$ , so (EVAL APPL RAISE3) gives  $E'[R] L \Downarrow_{w w'}^{s @ s'} \text{fail}$ , as required.
  - If  $E'[N] L \Downarrow_w^{s'} G$  was derived with (EVAL APPL RAISE1), then  $G = \text{fail}$  and  $N'_1 \Downarrow_w^{s'} c$ . By induction hypothesis,  $E'[R] \Downarrow_{w w'}^{s @ s'} c$ , so by (EVAL APPL RAISE1),  $E'[R] L \Downarrow_{w w'}^{s @ s'} \text{fail}$ .

□

Finally, we have all the ingredients to show that the small-step and the big-step sampling semantics both compute the same traces with the same weights.

**Theorem 1**  $M \Downarrow_w^s G$  if and only if  $(M, 1, s) \Rightarrow (G, w, [])$ .

PROOF.

$\Rightarrow$ : By induction on the derivation of  $M \Downarrow_w^s G$ . The most interesting case is definitely the following:

$$\frac{\text{(EVAL APPL)}}{\frac{M \Downarrow_{w_1}^{s_1} \lambda x.M' \quad N \Downarrow_{w_2}^{s_2} V \quad M'[V/x] \Downarrow_{w_3}^{s_3} G}{M N \Downarrow_{w_1 \cdot w_2 \cdot w_3}^{s_1 @ s_2 @ s_3} G}}$$

By induction hypothesis,  $(M, 1, s_1) \Rightarrow (\lambda x.M', w_1, [])$ ,  $(N, 1, s_2) \Rightarrow (V, w_2, [])$  and  $(M'[V/x], 1, s_3) \Rightarrow (G, w_3, [])$ . By Lemma 10 (for  $E = [] N$ ),  $(M N, 1, s_1) \Rightarrow ((\lambda x.M') N, w_1, [])$ . By Lemma 10 again (for  $E = (\lambda x.M') []$ ),  $((\lambda x.M') N, 1, s_2) \Rightarrow ((\lambda x.M') V, w_2, [])$ . By Lemma 19,  $(M N, 1, s_1 @ s_2) \Rightarrow ((\lambda x.M') V, w_1 w_2, [])$ . By (RED PURE),  $((\lambda x.M') V, w_1 \cdot w_2, []) \rightarrow (M'[V/x], w_1 \cdot w_2, [])$ , which implies  $(M N, 1, s_1 @ s_2) \Rightarrow ((\lambda x.M') V, w_1 w_2, [])$ . Thus, the desired result follows by Lemma 19.

- Case:  $\frac{\text{(EVAL VAL)}}{G \in \mathcal{GV}} \frac{G \in \mathcal{GV}}{G \Downarrow_1^[] G}$

Here,  $M = V$ ,  $w = 1$  and  $s = []$ . so  $(M, w_0, s_0)$  reduces to  $(V, w_0, s_0)$  in 0 steps by the small-step semantics.

- Case:  $\frac{\text{(EVAL RANDOM)}}{w = \text{pdf}_D(\vec{c}, c)} \frac{w > 0}{D(\vec{c}) \Downarrow_w^{[c]} c}$

By (RED RANDOM) (taking  $E = []$ ),  $(D(\vec{c}), 1, [c]) \rightarrow (c, w, [])$ .



- (EVAL RANDOM FAIL)

pdf<sub>D</sub>( $\vec{c}, c$ ) = 0

• Case:  $\frac{\text{pdf}_D(\vec{c}, c) = 0}{D(\vec{c}) \Downarrow_0^{[c]} \text{fail}}$

By (RED RANDOM FAIL) (taking  $E = []$ ),  $(D(\vec{c}), 1, [c]) \rightarrow (\text{fail}, 0, [])$ .

(EVAL PRIM)
- Case:  $\frac{g(\vec{c}) \Downarrow_1^{\parallel} \sigma_g(\vec{c})}{g(\vec{c}) \Downarrow_1^{\parallel} \sigma_g(\vec{c})}$

By (RED PURE) (taking  $E = []$ ),  $(g(\vec{c}), 1, []) \rightarrow (\sigma_g(\vec{c}), 1, [])$ .

(EVAL SCORE)
- $c \in (0, 1]$

• Case:  $\frac{\text{score}(c) \Downarrow_c^{\parallel} \text{true}}{\text{score}(c) \Downarrow_c^{\parallel} \text{true}}$

By (RED SCORE) (taking  $E = []$ ),  $(D(\vec{c}), 1, []) \rightarrow (c, w, [])$ .

(EVAL APPL)
- $M \Downarrow_{w_1}^{s_1} \lambda x.M'$

$N \Downarrow_{w_2}^{s_2} V$

• Case:  $\frac{M' [V/x] \Downarrow_{w_3}^{s_3} G}{M N \Downarrow_{w_1 \cdot w_2 \cdot w_3}^{s_1 @ s_2 @ s_3} G}$

By induction hypothesis,  $(M, 1, s_1) \Rightarrow (\lambda x.M', w_1, [])$ ,  $(N, 1, s_2) \Rightarrow (V, w_2, [])$  and  $(M' [V/x], 1, s_3) \Rightarrow (G, w_3, [])$ .

By Lemma 10 (for  $E = []$ ),  $(M N, 1, s_1) \Rightarrow ((\lambda x.M') N, w_1, [])$ .

By Lemma 10 again (for  $E = (\lambda x.M') []$ ),  $((\lambda x.M') N, 1, s_2) \Rightarrow ((\lambda x.M') V, w_2, [])$ .

By Lemma 19,  $(M N, 1, s_1 @ s_2) \Rightarrow ((\lambda x.M') V, w_1 w_2, [])$ .

By (RED PURE),  $((\lambda x.M') V, w_1 \cdot w_2, []) \rightarrow (M' [V/x], w_1 \cdot w_2, [])$ , which implies  $(M N, 1, s_1 @ s_2) \Rightarrow (M' [V/x], w_1 \cdot w_2, [])$ .

Thus, the desired result follows by Lemma 19.

(EVAL APPL RAISE1)
- $M \Downarrow_w^s \text{fail}$

• Case:  $\frac{M \Downarrow_w^s \text{fail}}{M N \Downarrow_w^s \text{fail}}$

By induction hypothesis,  $(M, 1, s) \Rightarrow (\text{fail}, w, [])$ .

By Lemma 11 (with  $E = []$ ),  $(M N, 1, s) \Rightarrow (\text{fail}, w, [])$ .

(EVAL APPL RAISE2)
- $M \Downarrow_w^s c$

• Case:  $\frac{M \Downarrow_w^s c}{M N \Downarrow_w^s \text{fail}}$

By induction hypothesis,  $(M, 1, s) \Rightarrow (c, w, [])$ . By Lemma 10 (with  $E = []$ ),  $(M N, 1, s) \Rightarrow (c N, w, [])$ .

By (RED PURE),  $(c N, w, []) \rightarrow (\text{fail}, w, [])$ .

Thus,  $(M N, 1, s) \Rightarrow (\text{fail}, w, [])$ .

(EVAL APPL RAISE3)
- $M \Downarrow_{w_1}^{s_1} \lambda x.M'$

$N \Downarrow_{w_2}^{s_2} \text{fail}$

• Case:  $\frac{N \Downarrow_{w_2}^{s_2} \text{fail}}{M N \Downarrow_{w_1 \cdot w_2}^{s_1 @ s_2} \text{fail}}$

By induction hypothesis,  $(M, 1, s_1) \Rightarrow (\lambda x.M', w_1, [])$ , and  $(N, 1, s_2) \Rightarrow (\text{fail}, w_2, [])$ .

By Lemma 10,  $(M N, 1, s_1) \Rightarrow ((\lambda x.M') N, w_1, [])$ .

By Lemma 11,  $((\lambda x.M') N, 1, s_2) \Rightarrow (\text{fail}, w_2, [])$ .

Thus, by Lemma 19,  $(M N, 1, s_1 @ s_2) \Rightarrow (\text{fail}, w_1 \cdot w_2, [])$ .

(EVAL IF TRUE)
- $M_2 \Downarrow_w^s G$

• Case:  $\frac{M_2 \Downarrow_w^s G}{\text{if true then } M_2 \text{ else } M_3 \Downarrow_w^s G}$

By (RED PURE) (taking  $E = []$ ),  $(\text{if true then } M_2 \text{ else } M_3, 1, s) \rightarrow (M_2, 1, s)$ . By induction hypothesis,  $(M_2, 1, s) \Rightarrow (G, w, [])$ .

Hence  $(\text{if 1 then } M_2 \text{ else } M_3, 1, s) \Rightarrow (G, w, [])$ .
- Case (EVAL IF FALSE): analogous to (EVAL IF TRUE)

(EVAL FAIL)

- Case:  $\frac{}{T \Downarrow_1^{\text{fail}}}$   
By (RED PURE),  $(T, 1, []) \rightarrow (\text{fail}, 1, [])$ .

$\Leftarrow$ : By induction on the length of the derivation of  $(M, 1, s) \Rightarrow (G, w, [])$ .

- Base case: If  $(M, 1, s) = (G, w, [])$ , then  $M \Downarrow_s^w G$  by (EVAL VAL).
- Induction step: assume  $(M, 1, s) \rightarrow (M', w', s') \rightarrow^n (G, w, [])$ . If  $(M, 1, s) \rightarrow (M', w', s')$  was derived with (RED RANDOM FAIL), then  $M = E[D(\vec{c})]$ ,  $n = 1$ ,  $s = [c]$ ,  $G = \text{fail}$  and  $w = w' = \text{pdf}_D(\vec{c}, c) = 0$ . By Lemma 21, we have  $M \Downarrow_0^{|c|} \text{fail}$ , as required. Otherwise, by Lemma 13,  $w' > 0$ , so by Lemma 17,  $(M', 1, s') \rightarrow^n (G, w/w', [])$ . By induction hypothesis,  $M' \Downarrow_{w/w'}^{s'} G$ . By Lemma 16,  $(M, 1, s^*) \rightarrow (M', w', [])$ , where  $s = s^* @ s'$ . Therefore, by Lemma 22,  $M \Downarrow_w^{s^* @ s'} G$ , and so  $M \Downarrow_w^s G$ .  $\square$

As a corollary of Theorem 1 and Lemma 7 we obtain:

**Lemma 23** *If  $M \Downarrow_w^s G$  and  $M \Downarrow_{w'}^{s'} G'$  then  $w = w'$  and  $G = G'$ .*

At this point, we have defined intuitive operational semantics based on the consumption of an explicit trace of randomness, we have defined no probability distributions. In the rest of this section we show that this semantics indeed associates a sub-probability distribution with each term. Before proceeding, however, we need some measure theory.

### 3.2 Some Measure-Theoretic Preliminaries.

We begin by recapitulating some standard definitions for sub-probability distributions and kernels over metric spaces. A  $\sigma$ -algebra (over a set  $X$ ) is a set  $\Sigma$  of subsets of  $X$  that contains  $\emptyset$ , and is closed under complement and countable union (and hence is closed under countable intersection). Let the  $\sigma$ -algebra *generated* by  $S$ , written  $\sigma(S)$ , be the set  $\sigma(S) \subseteq \mathcal{P}(X)$ , that is the least  $\sigma$ -algebra over  $\cup S$  that is a superset of  $S$ . In other words,  $\sigma(S)$  is the least set such that:

1. we have  $S \subseteq \sigma(S)$  and  $\emptyset \in \sigma(S)$ ; and
2.  $(\cup S) \setminus A \in \sigma(S)$  if  $A \in \sigma(S)$ ; and
3.  $\cup_{i \in \mathbb{N}} A_i \in \sigma(S)$  if each  $A_i \in \sigma(S)$ .

An equivalent definition is that  $\sigma(S) \triangleq \bigcap \{ \Sigma \mid S \subseteq \Sigma \text{ and } \Sigma \text{ is a } \sigma\text{-algebra} \}$ .

We write  $\mathbb{R}_+$  for  $[0, \infty]$  and  $\mathbb{R}_{[0,1]}$  for the interval  $[0, 1]$ . A *metric space* is a set  $X$  with a symmetric *distance function*  $\delta : X \times X \rightarrow \mathbb{R}_+$  that satisfies the triangle inequality  $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$  and the axiom  $\delta(x, x) = 0$ . We write  $\mathbf{B}(x, r) \triangleq \{y \mid \delta(x, y) < r\}$  for the open ball around  $x$  of radius  $r$ . We equip  $\mathbb{R}_+$  and  $\mathbb{R}_{[0,1]}$  with the standard metric  $\delta(x, y) = |x - y|$ , and products of metric spaces with the Manhattan metric (e.g.,  $\delta((x_1, x_2), (y_1, y_2)) = \delta(x_1, y_1) + \delta(x_2, y_2)$ ). The *Borel  $\sigma$ -algebra* on a metric space  $(X, \delta)$  is  $\mathcal{B}(X, \delta) \triangleq \sigma(\{\mathbf{B}(x, r) \mid x \in X \wedge r > 0\})$ . We often omit the arguments to  $\mathcal{B}$  when they are clear from the context.

A *measurable space* is a pair  $(X, \Sigma)$  where  $X$  is a set of possible outcomes, and  $\Sigma \subseteq \mathcal{P}(X)$  is a  $\sigma$ -algebra. As an example, we consider the extended positive real numbers  $\mathbb{R}_+$  equipped with the Borel  $\sigma$ -algebra  $\mathcal{R}$  (with respect to the standard metric) is the set  $\sigma(\{(a, b) \mid a, b \geq 0\})$ , which is the smallest  $\sigma$ -algebra containing all open (and closed) intervals. We can create finite products of measure spaces by iterating the construction  $(X, \Sigma) \times (X', \Sigma') = (X \times X', \sigma(A \times B \mid A \in \Sigma \wedge B \in \Sigma'))$ . If  $(X, \Sigma)$  and  $(X', \Sigma')$  are measurable spaces, then the function  $f : X \rightarrow X'$  is *measurable* if and only if for all  $A \in \Sigma'$ ,  $f^{-1}(A) \in \Sigma$ , where the *inverse image*  $f^{-1} : \mathcal{P}(X') \rightarrow \mathcal{P}(X)$  is given by  $f^{-1}(A) \triangleq \{x \in X \mid f(x) \in A\}$ .

A *measure*  $\mu$  on  $(X, \Sigma)$  is a function from  $\Sigma$  to  $\mathbb{R}_+$ , that is (1) zero on the empty set, that is,  $\mu(\emptyset) = 0$ , and (2) countably additive, that is,  $\mu(\cup_i A_i) = \sum_i \mu(A_i)$  if  $A_1, A_2, \dots$  are pair-wise

disjoint. The measure  $\mu$  is called a *probability measure* if  $\mu(X) = 1$ , and a *sub-probability measure* if  $\mu(X) \leq 1$ . If  $\mu$  is a measure and  $c \geq 0$ , we write  $c \cdot \mu$  for the measure  $A \mapsto c \cdot \mu(A)$ . For any element  $x$  of  $X$ , the Dirac measure  $\delta(x)$  is defined as follows:

$$\delta(M)(A) = \begin{cases} 1 & \text{if } x \in A; \\ 0 & \text{otherwise} \end{cases}$$

A *measure space* is a triple  $\mathcal{M} = (X, \Sigma, \mu)$  where  $\mu$  is a measure on the measurable space  $(X, \Sigma)$ . Given a measurable function  $f : X \rightarrow \mathbb{R}_+$ , the *integral* of  $f$  over  $\mathcal{M}$  can be defined following Lebesgue's theory and denoted as either of

$$\int f d\mu = \int f(x) \mu(dx) \in \mathbb{R}_+.$$

If  $A \in \Sigma$  we write  $1_A$  for the function that is 1 on  $A$  and 0 elsewhere. We then write

$$\int_A f d\mu \triangleq \int f(x) \cdot 1_A(x) \mu(dx).$$

We equip some measurable spaces  $(X, \Sigma)$  with a *stock measure*  $\mu$ . When  $f$  is measurable  $f : X \rightarrow \mathbb{R}_+$  we write  $\int f(s) ds$  (or shorter,  $\int f$ ) for  $\int f d\mu$ . In particular, we let the stock measure on  $(\mathbb{R}^n, \mathcal{B})$  be the Lebesgue measure.

### 3.3 Measure Space on Program Traces

In this section, we construct a measure space on the set  $\mathbb{S}$  of program traces: (1) we define a measurable space  $(\mathbb{S}, \mathcal{S})$  and (2) we equip it with a stock measure  $\mu$  to obtain our measure space  $(\mathbb{S}, \mathcal{S}, \mu)$ .

**The measurable space of program traces** To define the semantics of a program as a measure on the space of random choices, we first need to define a measurable space of program traces. Since a program trace is a sequence of random real variables of an arbitrary length, the set of all program traces is  $\mathbb{S} = \biguplus_{n \in \mathbb{N}} \mathbb{R}^n$ . Now, let us define the  $\sigma$ -algebra  $\mathcal{S}$  on  $\mathbb{S}$  as follows: let  $\mathcal{R}^n$  be the Borel  $\sigma$ -algebra on  $\mathbb{R}^n$ . Consider the class of sets  $\mathcal{S}$  of the form:

$$A = \biguplus_{n \in \mathbb{N}} H_n$$

where  $H_n \in \mathcal{R}^n$  for all  $n$ . Then  $\mathcal{S}$  is a  $\sigma$ -algebra.

**Lemma 24**  $\mathcal{S}$  is a  $\sigma$ -algebra on  $\mathbb{S}$ .

PROOF. We have  $\mathbb{S} = \biguplus_{n \in \mathbb{N}} \mathbb{R}^n$  and  $\mathbb{R}^n \in \mathcal{R}^n$  for all  $n$ , so  $\mathbb{S} \in \mathcal{S}$ .

If  $A$  is defined as above, then  $\mathbb{S} - A = \biguplus_{n \in \mathbb{N}} (\mathbb{R}^n - H_n)$ , where  $\mathbb{R}^n - H_n \in \mathcal{R}^n$  for all  $n$ , so  $\mathbb{S} - A \in \mathcal{S}$ .

For closure under countable union, take  $A_i = \biguplus_{n \in \mathbb{N}} H_{in}$  for all  $i \in \mathbb{N}$ , where  $H_{in} \in \mathcal{R}^n$  for all  $i, n$ . Then  $\bigcup_{i \in \mathbb{N}} A_i = \biguplus_{i \in \mathbb{N}} \biguplus_{n \in \mathbb{N}} H_{in} = \biguplus_{n \in \mathbb{N}} (\bigcup_{i \in \mathbb{N}} H_{in}) \in \mathcal{S}$ , because  $\bigcup_{i \in \mathbb{N}} H_{in} \in \mathcal{R}^n$ .

Thus,  $\mathcal{S}$  is a  $\sigma$ -algebra on  $\mathbb{S}$ . □

Since  $\mathcal{S}$  is a  $\sigma$ -algebra on  $\mathbb{S}$  it follows that  $(\mathbb{S}, \mathcal{S})$  is a measurable space.

**Stock Measure on Program Traces** Since each primitive distribution  $D$  has a density, the probability of each random value (and thus of each trace of random values) is zero. Instead, we define the trace and transition probabilities in terms of densities, with respect to the stock measure  $\mu$  on  $(\mathbb{S}, \mathcal{S})$  defined below.

$$\mu \left( \biguplus_{n \in \mathbb{N}} H_n \right) = \sum_{n=0}^{\infty} \lambda_n(H_n)$$

where  $\lambda_n$  is the Lebesgue measure on  $\mathbb{R}^n$ .

**Lemma 25**  $\mu$  is a measure on  $(\mathbb{S}, \mathcal{S})$ .

PROOF. We check the three properties:

1. Since for all  $n \in \mathbb{N}$  and  $H_n \in \mathcal{R}^n$ , we have  $\lambda_n(H_n) \in [0, \infty]$ , obviously  $\mu(\biguplus_{n \in \mathbb{N}} H_n) = \sum_{n=1}^{\infty} \lambda_n(H_n) \in [0, \infty]$
2. If  $H = \biguplus_{n \in \mathbb{N}} H_n = \emptyset$ , then  $H_n = \emptyset$  for all  $n$ , so  $\mu(H) = \sum_{n=1}^{\infty} \lambda_n(\emptyset) = 0$ .
3. Countable additivity: if  $H_1 = \biguplus_{n \in \mathbb{N}} H_{1n}, H_2 = \biguplus_{n \in \mathbb{N}} H_{2n}, \dots$  is a sequence of disjoint sets in  $\mathcal{S}$ , then:

$$\begin{aligned}
\mu\left(\biguplus_{m=1}^{\infty} H_m\right) &= \mu\left(\biguplus_{m=1}^{\infty} \biguplus_{n=0}^{\infty} H_{mn}\right) \\
&= \mu\left(\biguplus_{n=0}^{\infty} \biguplus_{m=1}^{\infty} H_{mn}\right) \\
&= \sum_{n=0}^{\infty} \lambda_n\left(\biguplus_{m=1}^{\infty} H_{mn}\right) \\
&= \sum_{n=0}^{\infty} \sum_{m=1}^{\infty} \lambda_n(H_{mn}) \\
&= \sum_{m=1}^{\infty} \sum_{n=0}^{\infty} \lambda_n(H_{mn}) \\
&= \sum_{m=1}^{\infty} \mu(H_m)
\end{aligned}$$

where the equality  $\sum_{n=0}^{\infty} \sum_{m=1}^{\infty} \lambda_n(H_{mn}) = \sum_{m=1}^{\infty} \sum_{n=0}^{\infty} \lambda_n(H_{mn})$  follows from Tonelli's theorem for series (see [28]).

□

A measure  $\mu$  on  $(X, \Sigma)$  is  $\sigma$ -finite if  $X = \bigcup_i A_i$  for some countable (finite or infinite) sequence of sets  $A_i \in \Sigma$  such that  $\mu(A_i) < \infty$ . If  $\mu$  is a  $\sigma$ -finite measure on  $(X, \Sigma)$ , the measure space  $(X, \Sigma, \mu)$  is also called  $\sigma$ -finite.  $\sigma$ -finite measure spaces behave better with respect to integration than those who are not.

In the following, let  $[a, b]^n = \{(x_1, \dots, x_n) \mid x_i \in [a, b] \ \forall i \in 1..n\}$ .

**Lemma 26** The measure  $\mu$  on  $(\mathbb{S}, \mathcal{S})$  is  $\sigma$ -finite.

PROOF. For every  $n \in \mathbb{N}$ , we have that  $\mathbb{R}^n = \bigcup_{k \in \mathbb{N}} [-k, k]^n$ . Hence,  $\mathbb{S} = \biguplus_{n \in \mathbb{N}} \mathbb{R}^n = \biguplus_{n \in \mathbb{N}} \bigcup_{k \in \mathbb{N}} [-k, k]^n$  is a countable union of sets in  $\mathcal{S}$  of the form  $[-k, k]^n$ . Finally, for all  $k, n \in \mathbb{N}$  we have  $\mu([-k, k]^n) = \lambda_n([-k, k]^n) = (2k)^n < \infty$ . □

It follows that  $(\mathbb{S}, \mathcal{S}, \mu)$  is a  $\sigma$ -finite measure space.

### 3.4 Distribution $\llbracket M \rrbracket_{\mathbb{S}}$ Induced by Sampling-Based Semantics

We define  $\mathcal{M}$  to be the set of Borel-measurable sets of terms equipped with the recursively defined metric  $d$  in Figure 5. The probability density of termination of a closed term  $M$  on a given trace is defined as follows.

$$\mathbf{P}_M(s) = \begin{cases} w & \text{if } M \Downarrow_w^s G \text{ for some } G \\ 0 & \text{otherwise} \end{cases}$$

**Lemma 27** For any closed term  $M$ , the function  $\mathbf{P}_M$  is measurable  $\mathbb{S} \rightarrow \mathbb{R}_+$ .

$$\begin{aligned}
d(x, x) &= 0 \\
d(c, d) &= |c - d| \\
d(MN, LP) &= d(M, L) + d(N, P) \\
d(g(V_1, \dots, V_n), g(W_1, \dots, W_n)) &= d(V_1, W_1) + \dots + d(V_n, W_n) \\
d(\lambda x.M, \lambda x.N) &= d(M, N) \\
d(D(V_1, \dots, V_n), D(W_1, \dots, W_n)) &= d(V_1, W_1) + \dots + d(V_n, W_n) \\
d(\text{score}(V), \text{score}(W)) &= d(V, W) \\
d(\text{if } V \text{ then } M \text{ else } N, \text{if } W \text{ then } L \text{ else } P) &= d(V, W) + d(M, L) + d(N, P) \\
d(\text{fail}, \text{fail}) &= 0 \\
d(M, N) &= \infty \text{ otherwise}
\end{aligned}$$

Figure 5: Metric  $d$  on terms.

PROOF. See Appendix A. □

This density function induces a sub-probability measure  $\llbracket M \rrbracket_{\Downarrow}^{\mathbb{S}}$  on closed terms by  $\llbracket M \rrbracket_{\Downarrow}^{\mathbb{S}}(A) = \int_A \mathbf{P}_M$ .

**Lemma 28** *The function  $\llbracket M \rrbracket_{\Downarrow}^{\mathbb{S}}$  is a measure on  $(\mathbb{S}, \mathcal{S})$ .*

PROOF. Since  $\mathbf{P}_M$  is a non-negative  $\mathcal{S}$ -measurable function,  $\llbracket M \rrbracket_{\Downarrow}^{\mathbb{S}}(A) = \int_A \mathbf{P}_M$  is a measure definition. □

The return value of a closed term  $M$  on a given trace is

$$\mathbf{O}_M(s) = \begin{cases} G & \text{if } M \Downarrow_w^s G \text{ for some } w \\ \text{fail} & \text{otherwise} \end{cases}$$

**Lemma 29** *For each  $M$ , the function  $\mathbf{O}_M$  is measurable  $\mathbb{S} \rightarrow \mathcal{GV}$ .*

PROOF. See Appendix A. □

The sampling-based semantics then induces a sub-distribution over generalised values  $\llbracket M \rrbracket_{\mathbb{S}}$  defined by

$$\llbracket M \rrbracket_{\mathbb{S}}(A) := \llbracket M \rrbracket_{\Downarrow}^{\mathbb{S}}(\mathbf{O}_M^{-1}(A)) = \int \mathbf{P}_M(s) \cdot \delta(\mathbf{O}_M(s))(A) ds.$$

We are using the exception **fail** for conditioning. To restrict attention to normal termination, we modify  $\mathbf{P}_M$  as follows.

$$\mathbf{P}_M^{\vee}(s) = \begin{cases} w & \text{if } M \Downarrow_w^s V \text{ for some } V \\ 0 & \text{otherwise.} \end{cases}$$

**Lemma 30**  *$\mathbf{P}_M^{\vee}(s)$  is measurable.*

PROOF. See Appendix A. □

As above, this density function generates sub-distributions over traces and values as, respectively

$$\begin{aligned}\llbracket M \rrbracket_{\Downarrow}^{\mathbb{S}\mathcal{V}}(A) &:= \llbracket M \rrbracket_{\Downarrow}^{\mathbb{S}}(A \cap \mathbf{O}_M^{-1}(\mathcal{V})) = \int_A \mathbf{P}_M^{\mathcal{V}}(s) ds \\ \llbracket M \rrbracket_{\mathbb{S}}^{\mathcal{V}}(A) &= \llbracket M \rrbracket_{\mathbb{S}}(A \cap \mathcal{V}) = \int \mathbf{P}_M^{\mathcal{V}}(s) \cdot \delta(\mathbf{O}_M(s))(A) ds\end{aligned}$$

## 4 Distribution-Based Operational Semantics

In this section, we will introduce distribution-based operational semantics in two flavours, and we prove them to be equivalent. Moreover, we will prove how all this is also equivalent to the sampling-based operational semantics from Section 3.

A term  $M$  is said to be *skeleton* iff no real number occurs in  $M$ , and each variable occurs *at most once* in  $M$ . The set of skeletons is  $\mathbf{SK}$ . Any closed term  $M$  can be written as  $N\{\vec{c}/\vec{x}\}$ , where  $N$  is a skeleton. The set of closed terms corresponding this way to a skeleton  $M \in \mathbf{SK}$  is denoted as  $\mathbf{TM}(M)$ . If the underlying term is a skeleton, substitution can be defined also when the substituted terms are *sets* of values rather than mere values, because variables occurs at most once; in that case, we will use the notation  $M\{X/x\}$ , where  $X$  is any set of values.

### 4.1 Subprobability Kernels

If  $(X, \Sigma)$  and  $(Y, \Sigma')$  are measurable spaces, then a function  $Q : X \times \Sigma' \rightarrow \mathbb{R}_{[0,1]}$  is called a (*subprobability*) *kernel* (from  $(X, \Sigma)$  to  $(Y, \Sigma')$ ) if

1. for every  $x \in X$ ,  $Q(x, \cdot)$  is a subprobability measure on  $(Y, \Sigma')$ ; and
2. for every  $A \in \Sigma'$ ,  $Q(\cdot, A)$  is a non-negative measurable function  $X \rightarrow \mathbb{R}_{[0,1]}$ .

$Q$  is said to be a *probability kernel* if  $Q(x, Y) = 1$  for all  $x \in X$ . When  $Q$  is a kernel, note that  $\int f(y) Q(x, dy)$  denotes the integral of  $f$  with respect to the measure  $Q(x, \cdot)$ .

Kernels can be composed in the following ways: If  $Q_1$  is a kernel from  $(X_1, \Sigma_1)$  to  $(X_2, \Sigma_2)$  and  $Q_2$  is a kernel from  $(X_2, \Sigma_2)$  to  $(X_3, \Sigma_3)$ , then  $Q_2 \circ Q_1 : (x, A) \mapsto \int Q_2(y, A) Q_1(x, dy)$  is a kernel from  $(X_1, \Sigma_1)$  to  $(X_3, \Sigma_3)$ . Moreover, if  $Q_1$  is a kernel from  $(X_1, \Sigma_1)$  to  $(X_2, \Sigma_2)$  and  $Q_2$  is a kernel from  $(X'_1, \Sigma'_1)$  to  $(X'_2, \Sigma'_2)$ , then  $Q_1 \times Q_2 : ((x, y), (A \times B)) \mapsto Q_1(x, A) \cdot Q_2(y, B)$  uniquely extends to a kernel from  $(X_1, \Sigma_1) \times (X'_1, \Sigma'_1)$  to  $(X_2, \Sigma_2) \times (X'_2, \Sigma'_2)$ .

The function  $f$  is a *density* of a measure  $\mu$  (with respect to the measure  $\nu$ ) if  $\mu(A) = \int_A f d\nu$  for all measurable  $A$ . Similarly, if  $Q : X \times \Sigma' \rightarrow \mathbb{R}_+$  is a kernel then  $q : X \times (Y) \rightarrow \mathbb{R}_+$  is said to be a density of  $Q$  if  $Q(v, A) = \int_A q(v, y) \nu(dy)$  for all  $A \in \Sigma$  and  $v \in X$ .

### 4.2 Approximation Small-Step Semantics

The first thing we need to do is to generalize deterministic reduction into a relation between closed terms and *distributions*. If  $\mu$  is a measure on terms, we let  $E\{\mu\}$  be the push-forward measure  $A \mapsto \mu(\{M \mid E[M] \in A\})$ .

*One-step evaluation* is a relation  $M \rightarrow \mathcal{D}$  between closed terms  $M$  and sub-probability measures  $\mathcal{D}$  on terms, defined as follows:

$$\begin{aligned}E[D(\vec{c})] &\rightarrow E\{\mu_{D(\vec{c})}\} \\ E[M] &\rightarrow \delta(E[N]) \text{ if } M \xrightarrow{\text{det}} N \\ E[\text{score}(c)] &\rightarrow c \cdot \delta(E[\text{true}]) \text{ if } 0 < c \leq 1\end{aligned}$$

We first of all want to show that one-step reduction is essentially deterministic, and that we have a form of deadlock-freedom.

**Lemma 31** *For every closed term  $M$ , either  $M$  is a generalized value or there is a unique  $\mathcal{D}$  such that  $M \rightarrow \mathcal{D}$ .*

PROOF. An easy consequence of Lemma 1.  $\square$

We need to prove the just introduced notion of one-step reduction to support composition. This is captured by the following result.

**Lemma 32**  *$\rightarrow$  is a subprobability kernel.*

Let  $R\Lambda_P = \{E[R] \in C\Lambda\}$  be the set of all closed reducible terms.

PROOF. Lemma 31 already tells us that  $\rightarrow$  can be seen as a function  $\rightarrow$  defined as follows:

$$\rightarrow(M, A) = \begin{cases} \mathcal{D}(A) & \text{if } M \rightarrow \mathcal{D}; \\ 0 & \text{otherwise.} \end{cases}$$

The fact that  $\rightarrow(M, \cdot)$  is a subprobability measure is easily verified. On other hand, the fact that  $\rightarrow(\cdot, A)$  is measurable amounts to proving that  $OS(A, B) = (\rightarrow(\cdot, A))^{-1}(B)$  is a measurable set of terms whenever  $B$  is a measurable set of real numbers. We will do that by showing that for every skeleton  $N$ , the set  $OS(A, B) \cap \mathbf{TM}(N)$  is measurable. The thesis then follows by observing that

$$OS(A, B) = \bigcup_{N \in \mathbf{SK}} OS(A, B) \cap \mathbf{TM}(N)$$

and that  $\mathbf{SK}$  is countable. Now, let us observe that for every skeleton  $N$ , the nature of any term  $L$  in  $\mathbf{TM}(N)$  as for if being a value, or containing a deterministic redex, or containing a sampling redex, only depends on  $N$  and not on the term  $L$ . As an example, terms in  $\mathbf{TM}(xy)$  are nothing but deterministic redexes (actually, all of them rewrites deterministically to  $\delta(\mathbf{fail})$ ). This allows us to proceed by distinguishing three cases:

- If all terms in  $\mathbf{TM}(N)$  are values, then it can be easily verified that

$$OS(A, B) \cap \mathbf{TM}(N) = \begin{cases} \mathbf{TM}(N) & \text{if } 0 \in B; \\ \emptyset & \text{if } 0 \notin B. \end{cases}$$

Both when  $0 \in B$  and when  $0 \notin B$ , then,  $OS(A, B) \cap \mathbf{TM}(N)$  is indeed measurable.

- If all terms in  $\mathbf{TM}(N)$  contain deterministic redexes, then

$$OS(A, B) \cap \mathbf{TM}(N) = \begin{cases} \rightarrow^{-1}(A) \cap \mathbf{TM}(N) & \text{if } 1 \in B \\ \emptyset & \text{if } 1 \notin B. \end{cases}$$

Since deterministic reduction  $\rightarrow$  is known to be measurable, then both when  $1 \in B$  and when  $1 \notin B$ , the set  $OS(A, B) \cap \mathbf{TM}(N)$  is measurable.

- The hardest case is when  $N$  is of the form  $G[\mathbf{D}(\vec{x})]$ , where  $G$  is an evaluation context. In this case, however, we can proceed by decomposing the function we want to prove measurable into three measurable functions:
  - The function  $app : \mathcal{C} \times C\Lambda \rightarrow C\Lambda$ , that given an evaluation context  $E$  and a term  $M$ , returns the term  $E[M]$ . This is proved measurable in the Appendix.
  - The function  $deapp : R\Lambda_P \rightarrow \mathcal{C} \times C\Lambda$  that “splits” a term in  $R\Lambda_P$  into an evaluation context  $E$  and a closed term  $M$ . This is proved measurable in the Appendix.
  - For every distribution identifier  $\mathbf{D}$ , the function  $distapp_{\mathbf{D}} : \mathbb{R}^n \rightarrow C\Lambda$  (where  $n$  is the arity of  $\mathbf{D}$ ) that, given a tuple of real numbers  $x$ , returns the term  $\mathbf{D}(x)$ . This function is a continuous function between two metric spaces, so measurable.
  - We know that for every distribution identifier  $\mathbf{D}$ , there is a kernel  $\mu_{\mathbf{D}} : \mathbb{R}^n \times \Sigma_{\mathbb{R}} \rightarrow \mathbb{R}_{[0,1]}$ . Moreover, one can also consider the Dirac kernel on evaluation contexts, namely  $I : \mathcal{C} \times \Sigma_{\mathcal{C}} \rightarrow \mathbb{R}_{[0,1]}$  where  $I(E, A) = \delta(E)(A)$ . Then, the product  $\mu_{\mathbf{D}} \times I$  is also a kernel, so measurable.

$$\boxed{
\begin{array}{c}
\frac{n > 0}{G \rightarrow_n \delta(G)} \quad \frac{}{M \rightarrow_0 \mathbf{0}} \\
\frac{M \rightarrow \mathcal{D} \quad \{N \rightarrow_n \mathcal{E}_N\}_{N \in \text{supp}(\mathcal{D})}}{M \rightarrow_{n+1} A \mapsto \int \mathcal{E}_N(A) \mathcal{D}(dN)}
\end{array}
}$$

Figure 6: Step-Indexed Approximation Small-Step Semantics.

This concludes the proof.  $\square$

Given a family  $\{\mathcal{D}_M\}_{M \in A}$  of  $A$ -subdistributions indexed by terms in a measurable set  $B$ , and a measurable set  $A$  of terms from  $A$ , we often write, with an abuse of notation,  $\mathcal{D}_M(A)$  for the function that assigns to any term  $M \in A$  the real number  $\mathcal{D}_M(A)$ . The *step-indexed approximation small-step semantics* is the family of  $n$ -indexed relations  $M \rightarrow_n \mathcal{D}$  between terms and distributions inductively defined in Figure 6. Since generalised values have no transitions (there is no  $\mathcal{D}$  such that  $G \rightarrow \mathcal{D}$ ), the rules above are disjoint and so there is at most one  $\mathcal{D}$  such that  $M \rightarrow_n \mathcal{D}$ .

**Lemma 33** *For every  $n \in \mathbb{N}$ , the function  $\rightarrow_n$  is a kernel.*

PROOF. By induction on  $n$ :

- $\rightarrow_0$  can be seen as the function  $\hat{\rightarrow}_0$  that attributes 0 to any pair  $(M, A)$ . This is clearly a kernel.
- $\rightarrow_{n+1}$  can be seen as the function  $\hat{\rightarrow}_{n+1}$  defined as follows:

$$\hat{\rightarrow}_{n+1}(M, A) = \begin{cases} 1 & \text{if } M \in \mathcal{GV} \text{ and } M \in A; \\ 0 & \text{if } M \in \mathcal{GV} \text{ and } M \notin A; \\ (\int \hat{\rightarrow}_n(N, A) \mathcal{D}(dN)) & \text{if } M \rightarrow \mathcal{D}. \end{cases}$$

The fact that  $\hat{\rightarrow}_{n+1}(M, \cdot)$  is a measure for every  $M$  is clear, and can be proved by case distinction on  $M$ . On the other hand, if  $B$  is a measurable set of reals, then:

$$\begin{aligned}
(\hat{\rightarrow}_{n+1}(\cdot, A))^{-1}(B) &= (\hat{\rightarrow}_{n+1}(\cdot, A))^{-1}(B) \cap \mathcal{GV} \cup \\
&\quad (\hat{\rightarrow}_{n+1}(\cdot, A))^{-1}(B) \cap (C\Lambda - \mathcal{GV}).
\end{aligned}$$

Now, the fact that  $(\hat{\rightarrow}_{n+1}(\cdot, A))^{-1}(B) \cap \mathcal{GV}$  is a measurable set of terms is clear: it is  $A \cap \mathcal{GV}$  if  $1 \in B$  and  $\emptyset$  otherwise. But how about  $(\hat{\rightarrow}_{n+1}(\cdot, A))^{-1}(B) \cap (C\Lambda - \mathcal{GV})$ ? In that case, we just need to notice that

$$(\hat{\rightarrow}_{n+1}(\cdot, A))^{-1}(B) \cap (C\Lambda - \mathcal{GV}) = (\hat{\rightarrow}_n \circ \rightarrow)^{-1}(B)$$

where  $\hat{\rightarrow}_n$  and  $\rightarrow$  are kernels (the former by induction hypothesis, the latter by Lemma 32). Since kernels compose, this concludes the proof.  $\square$

**Lemma 34** *For every closed term  $M$  and for every  $n \in \mathbb{N}$  there is a unique distribution  $\mathcal{D}$  such that  $M \rightarrow_n \mathcal{D}$ .*

PROOF. This is an easy consequence of Lemma 33.  $\square$

### 4.3 Approximation Big-Step Semantics

Given a measurable subset of terms  $A$ ,  $\mathcal{M}^A$  is the *restriction* of  $\mathcal{M}$  to elements in  $A$ , i.e.,  $\mathcal{M}^A = \{B \cap A \mid B \in \mathcal{M}\}$ . Any pair in the form  $\mathcal{M}_A = (A, \mathcal{M}^A)$  is by construction a measurable space. Given a distribution  $\mathcal{D}$  and a measurable set  $A$ ,  $\mathcal{D}^A$  is the *restriction* of  $\mathcal{D}$  to elements in  $A$ , i.e.,  $\mathcal{D}^A(B) = \mathcal{D}(A \cap B)$ . This construction yields a measure space  $\mathcal{M}_A^{\mathcal{D}} = (A, \mathcal{M}^A, \mathcal{D}^A)$ . As a consequence, the restriction of  $\mathcal{D}$  to elements *not* in  $A$  is  $\mathcal{D}^{\mathcal{GV}-A}$ .



$$\boxed{
\begin{array}{c}
\frac{n > 0}{G \Downarrow_n \delta(G)} \quad \frac{}{M \Downarrow_0 \mathbf{0}} \quad \frac{n > 0}{T \Downarrow_n \delta(\mathbf{fail})} \\
\\
\frac{n > 0}{D(\vec{c}) \Downarrow_n \mu_D(\vec{c})} \quad \frac{n > 0}{g(\vec{c}) \Downarrow_n \delta(\sigma_g(\vec{c}))} \quad \frac{0 < c \leq 1 \quad n > 0}{\mathbf{score}(c) \Downarrow_n c \cdot \delta(\mathbf{true})} \\
\\
\frac{M \Downarrow_n \mathcal{D}}{\mathbf{if true then } M \mathbf{ else } N \Downarrow_{n+1} \mathcal{D}} \quad \frac{N \Downarrow_n \mathcal{D}}{\mathbf{if false then } M \mathbf{ else } N \Downarrow_{n+1} \mathcal{D}} \\
\\
\frac{M \Downarrow_n \mathcal{D} \quad N \Downarrow_n \mathcal{E} \quad \{L\{V/x\} \Downarrow_n \mathcal{E}_{L,V}\}_{(\lambda x.L) \in \text{supp}(\mathcal{D}), V \in \text{supp}(\mathcal{E})}}{MN \Downarrow_{n+1} A \mapsto \mathcal{D}^{\mathcal{E}}(A) + \mathcal{D}(\mathbb{R}) \cdot \delta(\mathbf{fail}) + \mathcal{D}(\mathcal{V}_\lambda) \cdot \mathcal{E}^{\mathcal{E}}(A) + \iint \mathcal{E}_{L,V}(A) \mathcal{D}^{\mathcal{V}_\lambda}(\lambda x.dL) \mathcal{E}^{\mathcal{V}}(dV)}
\end{array}
}$$

Figure 7: Step Indexed Approximation Big-Step Semantics.

Given a measurable set of terms  $A$ , an  $A$ -distribution is a sub-probability measure on  $A$ , that is, a measure  $\mathcal{D} : \mathcal{M}^A \rightarrow \mathbb{R}_{[0,1]}$  such that  $\mathcal{D}(A) \leq 1$ . We write  $\mathbf{0}$  for the zero distribution  $A \mapsto 0$ .

A *value distribution* is a  $\mathcal{V}$ -distribution, that is, a sub-probability measure  $\mathcal{D} : \mathcal{M}^{\mathcal{V}} \rightarrow \mathbb{R}_{[0,1]}$  such that  $\mathcal{D}(\mathcal{V}) \leq 1$ .

The *step-indexed approximation big-step semantics*  $M \Downarrow_n \mathcal{D}$  is the  $n$ -indexed family of relations inductively defined by the rules in Figure 7.

Above, the rule for applications is the most complex, with the resulting distribution consisting of three exceptional terms in addition to the normal case. To better understand this rule, one can study what happens if we replace general applications with a let construct plus application of values to values. Then we would end up having the following three rules, instead of the rule for application above:

$$\begin{array}{c}
\frac{M \Downarrow_n \mathcal{D} \quad \{N\{V/x\} \Downarrow_n \mathcal{E}_V\}_{V \in \text{supp}(\mathcal{D})}}{\mathbf{let } x = M \mathbf{ in } N \Downarrow_{n+1} A \mapsto \left( \begin{array}{l} \mathcal{D}^{\mathcal{E}}(A) + \mathcal{D}(\mathbb{R}) \cdot \delta(\mathbf{fail}) \\ + \int \mathcal{E}_V(A) \mathcal{D}^{\mathcal{V}}(dV) \end{array} \right)} \\
\\
\frac{M\{V/x\} \Downarrow_n \mathcal{E}}{(\lambda x.M)V \Downarrow_{n+1} \mathcal{E}} \quad \frac{n > 0}{c V \Downarrow_n \delta(\mathbf{fail})}
\end{array}$$

#### 4.4 Beyond Approximations

The set of value distributions with the pointwise order forms an  $\omega\mathbf{CPO}$ , and thus any denumerable, directed set of value distributions has a least upper bound. One can define the *small-step semantics* and the *big-step semantics* as, respectively, the two distributions

$$\begin{aligned}
\llbracket M \rrbracket_{\Rightarrow} &= \sup\{\mathcal{D} \mid M \rightarrow_n \mathcal{D}\} \\
\llbracket M \rrbracket_{\Downarrow} &= \sup\{\mathcal{D} \mid M \Downarrow_n \mathcal{D}\}
\end{aligned}$$

It would be quite disappointing if the two objects above were different. Indeed, this section is devoted to proving the following theorem:

**Theorem 2** *For every term  $M$ ,  $\llbracket M \rrbracket_{\Rightarrow} = \llbracket M \rrbracket_{\Downarrow}$ .*

The following is a fact which will be quite useful in the following:

**Lemma 35 (Monotonicity)** *If  $M \rightarrow_n \mathcal{D}$ ,  $m \geq n$  and  $M \rightarrow_m \mathcal{E}$ , then  $\mathcal{E} \geq \mathcal{D}$ .*

Theorem 2 can be proved by showing that any big-step approximation can itself over-approximated with small-step, and vice versa. Let us start by showing that, essentially, the big-step rule for applications is small-step-admissible:

**Lemma 36** *If  $M \rightarrow_n \mathcal{D}$ ,  $N \rightarrow_m \mathcal{E}$ , and for all  $L$  and  $V$ ,  $L\{V/x\} \rightarrow_p \mathcal{E}_{L,V}$ , then  $MN \rightarrow_{n+m+p} \mathcal{F}$  such that for all  $A$*

$$\begin{aligned} \mathcal{F}(A) \geq & \mathcal{D}^{\mathcal{E}}(A) + \mathcal{D}(\mathbb{R}) \cdot \delta(\mathbf{fail}) + \mathcal{D}(\mathcal{V}_\lambda) \cdot \mathcal{E}^{\mathcal{E}}(A) \\ & + \iint \mathcal{E}_{L,V}(A) \mathcal{D}^{\mathcal{V}_\lambda}(\lambda x.L) \mathcal{E}^{\mathcal{V}}(dV). \end{aligned}$$

PROOF. First of all, one can prove that if  $N \rightarrow_n \mathcal{D}$  and  $L\{V/x\} \rightarrow_m \mathcal{E}_V$  for all  $V$  then  $(\lambda x.L)N \rightarrow_{n+m} \mathcal{F}$  where  $\mathcal{F}(A) \geq \mathcal{D}^{\mathcal{E}}(A) + \int \mathcal{E}_V(A) \mathcal{D}^{\mathcal{V}}(dV)$  for all  $A$ . This is an induction on  $n$ .

- If  $n = 0$ , then  $\mathcal{D}$  is necessarily the zero distribution  $A \mapsto 0$ . Then  $\mathcal{F}(A) \geq 0 = \mathcal{D}^{\mathcal{E}}(A) + \int \mathcal{E}_V(A) \mathcal{D}^{\mathcal{V}}(dV)$ .
- Suppose the thesis holds for  $n$ , and let's try to prove the thesis for  $n + 1$ . We proceed by further distinguishing some subcases:
  - If  $N$  is a value  $W$ , then  $\mathcal{D} = \delta(W)$ ,  $\mathcal{D}^{\mathcal{E}}$  is the zero distribution and thus

$$(\lambda x.L)N \rightarrow_{m+1} (A \mapsto \mathcal{D}^{\mathcal{E}}(A) + \int \mathcal{E}_V(A) \mathcal{D}^{\mathcal{V}}(dV)).$$

The thesis follows by monotonicity.

- If  $N$  is an exception  $\mathbf{fail}$ , then  $\mathcal{D} = \delta(\mathbf{fail})$ , and since  $(\lambda x.L)\mathbf{fail} \rightarrow \mathbf{fail}$ , we can conclude that, since  $\mathcal{D}^{\mathcal{V}}$  is the zero distribution,

$$(\lambda x.L)N \rightarrow_2 \delta(\mathbf{fail}) = (A \mapsto \mathcal{D}^{\mathcal{E}}(A) + \int \mathcal{E}_V(A) \mathcal{D}^{\mathcal{V}}(dV)).$$

The thesis again follows by monotonicity.

- If  $N$  is not a generalized value, then, necessarily  $\mathcal{D}(A) = \int \mathcal{G}_P(A) \mathcal{H}(dP)$ , where  $N \rightarrow \mathcal{H}$  and  $P \rightarrow_n \mathcal{G}_P$  for every  $P$ . By induction hypothesis, there are measures  $\mathcal{J}_P$  such that  $(\lambda x.L)P \rightarrow_{n+m} \mathcal{J}_P$ , and, for all  $A$ ,

$$\mathcal{J}_P(A) \geq \mathcal{G}_P^{\mathcal{E}} + \int \mathcal{E}_V(A) \mathcal{G}_P^{\mathcal{V}}(dV)$$

Let now  $E$  be the evaluation context  $(\lambda x.L)[\cdot]$ . Then, it holds that  $(\lambda x.L)N \rightarrow E\{\mathcal{H}\}$  and thus:

$$(\lambda x.L)N \rightarrow_{n+m+1} (A \mapsto \int \mathcal{J}_P(A) (E\{\mathcal{H}\}((\lambda x.L)dP))).$$

We can now observe that:

$$\begin{aligned} \int \mathcal{J}_P(A) (E\{\mathcal{H}\}((\lambda x.L)dP)) &= \int \mathcal{J}_P(A) \mathcal{H}(dP) \\ &\geq \int \mathcal{G}_P^{\mathcal{E}}(A) \mathcal{H}(dP) + \iint \mathcal{E}_V(A) \mathcal{G}_P^{\mathcal{V}}(dV) \mathcal{H}(dP) \\ &= \mathcal{D}^{\mathcal{E}}(A) + \iint \mathcal{E}_V(A) \mathcal{G}_P^{\mathcal{V}}(dV) \mathcal{H}(dP) \\ &= \mathcal{D}^{\mathcal{E}}(A) + \int \mathcal{E}_V(A) \mathcal{D}^{\mathcal{V}}(dV). \end{aligned}$$

Then one can prove the statement of the lemma, again by induction on  $n$ , following the same strategy as above.  $\square$

**Lemma 37** *If  $M \Downarrow_n \mathcal{D}$ , then there are  $m, \mathcal{E}$  such that  $M \rightarrow_m \mathcal{E}$  and  $\mathcal{E} \geq \mathcal{D}$ .*

PROOF. By induction on  $n$ , we can prove that if  $M \Downarrow_n \mathcal{D}$ , then  $M \rightarrow_{3^n} \mathcal{E}$  where  $\mathcal{E} \geq \mathcal{D}$ . The only interesting case is the one in which  $M$  is an application, but in that case we can simply go via Lemma 36.  $\square$

At this point, we already know that  $\llbracket M \rrbracket_{\Rightarrow} \geq \llbracket M \rrbracket_{\Downarrow}$ . The symmetric inequality can be proved by showing that the big-step rule for applications can be *inverted* in the small-step:

**Lemma 38** *If  $MN \rightarrow_{n+1} \mathcal{D}$ , then  $M \rightarrow_n \mathcal{E}$ ,  $N \rightarrow_n \mathcal{F}$  and for all  $P$  and  $V$ ,  $P\{V/x\} \rightarrow_n \mathcal{G}_{P,V}$  such that for all  $A$ ,*

$$\begin{aligned} \mathcal{D}(A) &\leq \mathcal{E}^{\mathcal{E}}(A) + \mathcal{E}(\mathbb{R}) \cdot \delta(\mathbf{fail}) + \mathcal{E}(\mathcal{V}_{\lambda}) \cdot \mathcal{F}^{\mathcal{E}}(A) \\ &\quad + \iint \mathcal{G}_{P,V}(A) \mathcal{E}^{\mathcal{V}_{\lambda}}(\lambda x.dP) \mathcal{F}^{\mathcal{V}}(dV) \end{aligned}$$

PROOF. By induction on  $n$ .

- If  $n = 0$ , then  $\mathcal{D}$  is the zero distribution, and so are  $\mathcal{E}$ ,  $\mathcal{F}$  and all  $\mathcal{G}_{P,V}$ .
- Suppose the thesis holds for every natural number smaller than  $n$  and prove it for  $n$ . Let us distinguish a few cases, and examine the most relevant ones:
  - If  $M$  is an abstraction  $\lambda x.L$  and  $N$  is a value  $W$ , then  $M \rightarrow \delta(L\{W/x\})$  and  $L\{W/x\} \rightarrow_n \mathcal{D}$ . We can then observe that

$$\begin{aligned} \mathcal{E} &= \delta(\lambda x.L) \\ \mathcal{F} &= \delta(W) \\ \mathcal{G}_{P,V} &= \mathcal{D} \text{ whenever } P = L \text{ and } V = W \end{aligned}$$

Just observe that

$$\mathcal{D}(A) = \iint \mathcal{G}_{P,V}(A) \mathcal{E}(\lambda x.dP) \mathcal{F}(dV)$$

and that  $\mathcal{E}^{\mathcal{E}} = \mathcal{E}^{\mathbb{R}} = \mathcal{F}^{\mathcal{E}} = \mathbf{0}$ .

- If none of  $M$  and  $N$  are values, then  $M \rightarrow \mathcal{L}$  and thus  $MN \rightarrow E\{\mathcal{L}\}$  where  $E = [\cdot]N$ . Moreover,  $LN \rightarrow_n \mathcal{H}_L$ , where

$$\mathcal{D}(A) = \int \mathcal{H}_L(A) E\{\mathcal{L}\}((dL)N) = \int \mathcal{H}_L(A) \mathcal{L}(dL)$$

We apply the induction hypothesis (and monotonicity) to each of the  $LN \rightarrow_n \mathcal{H}_L$ , and we obtain that  $L \rightarrow_{n-1} \mathcal{I}_L$ ,  $N \rightarrow_n \mathcal{F}$  and  $P\{V/x\} \rightarrow_n \mathcal{G}_{P,V}$ , where

$$\begin{aligned} \mathcal{H}_L(A) &\leq \mathcal{I}_L^{\mathcal{E}}(A) + \mathcal{I}_L(\mathbb{R}) \cdot \delta(\mathbf{fail}) + \mathcal{I}_L(\mathcal{V}_{\lambda}) \cdot \mathcal{F}^{\mathcal{E}}(A) \\ &\quad + \iint \mathcal{G}_{P,V}(A) \mathcal{I}_L^{\mathcal{V}_{\lambda}}(\lambda x.dP) \mathcal{F}^{\mathcal{V}}(dV) \end{aligned}$$

Now let  $\mathcal{E}$  be the measure

$$A \mapsto \int \mathcal{I}_L(A) \mathcal{L}(dL).$$

Clearly,  $M \rightarrow_n \mathcal{E}$ . Moreover,

$$\begin{aligned} \mathcal{D}(A) &= \int \mathcal{H}_L(A) \mathcal{L}(dL) \\ &\leq \int \mathcal{I}_L^{\mathcal{E}}(A) \mathcal{L}(dL) + \int \mathcal{I}_L(\mathbb{R}) \cdot \delta(\mathbf{fail}) \mathcal{L}(dL) \\ &\quad + \int \mathcal{I}_L(\mathcal{V}_{\lambda}) \cdot \mathcal{F}^{\mathcal{E}}(A) \mathcal{L}(dL) \\ &\quad + \iiint \mathcal{G}_{P,V}(A) \mathcal{I}_L^{\mathcal{V}_{\lambda}}(\lambda x.dP) \mathcal{F}^{\mathcal{V}}(dV) \mathcal{L}(dL) \\ &= \mathcal{E}^{\mathcal{E}}(A) + \mathcal{E}(\mathbb{R}) \cdot \delta(\mathbf{fail}) + \mathcal{E}(\mathcal{V}_{\lambda}) \cdot \mathcal{F}^{\mathcal{E}}(A) \\ &\quad + \iint \mathcal{G}_{P,V}(A) \mathcal{E}^{\mathcal{V}_{\lambda}}(\lambda x.dP) \mathcal{F}^{\mathcal{V}}(dV) \end{aligned}$$

□

**Lemma 39** *If  $M \rightarrow_n \mathcal{D}$ , then there is  $\mathcal{E}$  such that  $M \Downarrow_n \mathcal{E}$  and  $\mathcal{E} \geq \mathcal{D}$ .*

PROOF. Again, this is an induction on  $n$  that makes essential use, this time, of Lemma 38. □

RESTATEMENT OF THEOREM 2 *For all  $M$ ,  $\llbracket M \rrbracket_{\Rightarrow} = \llbracket M \rrbracket_{\Downarrow}$ .*

PROOF. This is a consequence of Lemma 37 and Lemma 39. □

In subsequent sections we let  $\llbracket M \rrbracket$  stand for  $\llbracket M \rrbracket_{\Rightarrow}$  or  $\llbracket M \rrbracket_{\Downarrow}$ .

## 4.5 Distribution-based and Sampling-based Semantics are Equivalent

This section is a proof of Theorem 3.

**Theorem 3** *For every term  $M$ ,  $\llbracket M \rrbracket_{\mathbb{S}} = \llbracket M \rrbracket$ .*

The way to prove Theorem 3 is by looking at traces of *bounded* length. For every  $n \in \mathbb{N}$ , let  $\mathbb{S}_n$  be the set of sample traces of length at most  $n$ . By a similar construction as we showed in Section 3.3 it has itself the structure of a measure space. Let us then define the value distribution  $\llbracket M \rrbracket_{\mathbb{S}}^n$  as follows:

$$\llbracket M \rrbracket_{\mathbb{S}}^n(A) = \int_{\mathbb{S}_n} \mathbf{P}_M(s) \cdot \delta(\mathbf{O}_M(s))(A) ds$$

If  $\text{dom}(f) = \mathbb{S}$  we write  $f|_n$  for the restriction of  $f$  to  $\mathbb{S}_n$ . The integral over *all* traces can be seen as the limit of all integrals over bounded-length traces:

**Lemma 40**  *$\int f = \sup_n \int f|_n$  whenever  $f$  is measurable wrt. the stock measure on  $\mathbb{S}$ .*

PROOF. Let  $g_n$  be the function that is as  $f$  on  $\mathbb{S}_n$  and 0 outside. Then  $\int f|_n = \int g_n$ . Since the  $g_n$  are converging to  $f$  pointwise from below, we also have  $\int f = \sup_n \int g_n$  by the monotone convergence theorem. □

A corollary is that  $\llbracket M \rrbracket_{\mathbb{S}} = \sup_{n \in \mathbb{N}} \llbracket M \rrbracket_{\mathbb{S}}^n$ .

**Lemma 41** *If  $M \rightarrow \mathcal{D}$ , then  $\llbracket M \rrbracket = A \mapsto \int \llbracket N \rrbracket(A) \mathcal{D}(dN)$*

PROOF. For every  $n$  and for every term  $N$ , let  $\mathcal{E}_N^n$  be the unique value distribution such that  $N \rightarrow_n \mathcal{E}_N^n$ . By definition, we have that

$$\mathcal{E}_M^{n+1}(A) = \int \mathcal{E}_N^n(A) \mathcal{D}(dN).$$

By the monotone convergence theorem, then,

$$\begin{aligned} \llbracket M \rrbracket(A) &= \sup_n \mathcal{E}_M^{n+1}(A) = \sup_n \int \mathcal{E}_N^n(A) \mathcal{D}(dN) \\ &= \int (\sup_n \mathcal{E}_N^n(A)) \mathcal{D}(dN) = \int \llbracket N \rrbracket(A) \mathcal{D}(dN). \end{aligned}$$

□

The following is a useful technical lemma.

**Lemma 42**  $\llbracket E[\mathbf{D}(\vec{c})] \rrbracket_{\mathbb{S}}^{n+1}(A) = \int \llbracket N \rrbracket_{\mathbb{S}}^n(A) E\{\mu_{\mathbf{D}}(\vec{c})\}(dN).$

**Lemma 43** *If  $M \rightarrow \mathcal{D}$ , then  $\llbracket M \rrbracket_{\mathbb{S}} = A \mapsto \int \llbracket N \rrbracket_{\mathbb{S}}(A) \mathcal{D}(dN)$*

A program  $M$  is said to *deterministically diverge* iff  $(M, 1, []) \Rightarrow (N, w, s)$  implies  $w = 1$ ,  $s = []$  and  $N$  is *not* a generalized value. A program  $M$  is said to *deterministically converge to a program  $N$*  iff  $(M, 1, []) \Rightarrow (N, 1, [])$ . Terms that deterministically diverge all have a very predictable semantics, both distribution and sampling-based. Moreover, any term that deterministically converges to another term have the same semantics of the latter. This is captured by the following results.

**Lemma 44** *If  $M$  deterministically diverges then  $\llbracket M \rrbracket = \llbracket M \rrbracket_{\mathbb{S}} = \mathbf{0}$ .*

PROOF. One can easily prove, by induction on  $n$ , that if  $M$  deterministically diverges, then  $M \rightarrow_n \mathbf{0}$ :

- If  $n = 0$ , then  $M \rightarrow_n \mathbf{0}$  by definition.
- About the inductive case, since  $M$  cannot be a generalized value, it must be that  $M \rightarrow \delta(N)$  (where  $N$  deterministically diverges) and that  $M \rightarrow_{n+1} \mathcal{D}$ , where  $N \rightarrow_n \mathcal{D}$ . By induction hypothesis,  $\mathcal{D} = \mathbf{0}$ .

The fact that  $\llbracket M \rrbracket_{\mathbb{S}} = \mathbf{0}$  is even simpler to prove, since if  $M$  deterministically diverges, then there cannot be any  $s, w, V$  such that  $M \Downarrow_w^s V$ , and thus  $\mathbf{P}_M(s)$  is necessarily 0.  $\square$

**Lemma 45** *Let  $M$  be a term that deterministically converges to a term  $N$ . Then:*

- $\mathcal{D} \leq \mathcal{E}$  whenever  $M \rightarrow_n \mathcal{D}$ ; and  $N \rightarrow_n \mathcal{E}$ ;
- $\llbracket M \rrbracket_{\mathbb{S}}^n = \llbracket N \rrbracket_{\mathbb{S}}^n$ ;
- $\llbracket M \rrbracket = \llbracket N \rrbracket$  and  $\llbracket M \rrbracket_{\mathbb{S}} = \llbracket N \rrbracket_{\mathbb{S}}$

PROOF. The first point is an induction on the structure of the proof that  $M$  deterministically converge to  $N$ . Let us consider the second and third points. Since equality is transitive, we can assume, without losing any generality, that  $(M, 1, []) \rightarrow (N, 1, [])$ , namely that  $M \xrightarrow{\text{det}} N$ . With the latter hypothesis, it is easy to realize that  $M \Downarrow_s^w V$  iff  $N \Downarrow_s^w V$  and that  $M \rightarrow_{n+1} \mathcal{D}$  iff  $N \rightarrow_n \mathcal{D}$ . The thesis easily follows.  $\square$

**Lemma 46** *For every generalized value  $G$ , it holds that  $\llbracket G \rrbracket = \llbracket G \rrbracket_{\mathbb{S}} = \delta(G)$ .*

If a term does not diverge deterministically, then it converges either to a generalized value or to another term performing a sampling.

**Lemma 47** *For every program  $M$ , exactly one of the following conditions holds:*

- $M$  deterministically diverges;
- There is generalized value  $G$  such that  $M$  deterministically converges to  $G$
- There are  $E, D, c_1, \dots, c_{|D|}$  such that  $M$  deterministically converges to  $E[D(c_1, \dots, c_{|D|})]$ .

PROOF. Easy.  $\square$

We are finally ready to give the two main lemmas that lead to a proof of Theorem 3. The first one tells us that any distribution-based approximation is smaller than the sampling based semantics:

**Lemma 48** *If  $M \rightarrow_n \mathcal{D}$ , then  $\mathcal{D} \leq \llbracket M \rrbracket_{\mathbb{S}}$ .*

PROOF. By induction on  $n$ :

- If  $n = 0$ , then  $\mathcal{D}$  is necessarily  $\mathbf{0}$ , and we are done.
- About the inductive case, let's distinguish three cases depending on the three cases of Lemma 47, applied to  $M$ :
  - If  $M$  deterministically diverges, then by Lemma 44,  $\mathcal{D} \leq \llbracket M \rrbracket = \llbracket M \rrbracket_{\mathbb{S}}$ .
  - If  $M$  deterministically converges to a generalized value  $G$ , then by Lemma 45 and Lemma 46, it holds that

$$\mathcal{D} \leq \llbracket M \rrbracket = \llbracket G \rrbracket = \delta(G) = \llbracket G \rrbracket_{\mathbb{S}} = \llbracket M \rrbracket_{\mathbb{S}}.$$

- If  $M$  deterministically converges to  $E[D(\vec{c})]$ , let  $\mathcal{E}$  be such that  $E[D(\vec{c})] \rightarrow_{n+1} \mathcal{E}$ . By Lemma 45 and Lemma 42 we have, by induction hypothesis, that

$$\begin{aligned} \mathcal{D}(A) &\leq \mathcal{E}(A) = \int \mathcal{F}_N(A) E\{\mu_D(\vec{c})\}(dN) \\ &\leq \int \llbracket N \rrbracket_{\mathbb{S}}(A) E\{\mu_D(\vec{c})\}(dN) \\ &= \llbracket M \rrbracket_{\mathbb{S}} \end{aligned}$$

where  $N \rightarrow_n \mathcal{F}_N$ . □

The second main lemma tells us that if we limit our attention to traces of length at most  $n$ , then we stay below distribution-based semantics:

**Lemma 49** *For every  $n \in \mathbb{N}$ ,  $\llbracket M \rrbracket_{\mathbb{S}}^n \leq \llbracket M \rrbracket$ .*

PROOF. By induction on  $n$ :

- In the base case, then let us distinguish three cases depending on the three cases of Lemma 47, applied to  $M$ :
  - If  $M$  deterministically diverges, then by Lemma 44,  $\llbracket M \rrbracket_{\mathbb{S}}^0 \leq \llbracket M \rrbracket_{\mathbb{S}} = \llbracket M \rrbracket$ .
  - If  $M$  deterministically converges to a generalized value  $G$ , then by Lemma 45 and Lemma 46, it holds that

$$\llbracket M \rrbracket_{\mathbb{S}}^0 = \llbracket G \rrbracket_{\mathbb{S}}^0 \leq \llbracket G \rrbracket_{\mathbb{S}} = \delta(G) = \llbracket G \rrbracket = \llbracket M \rrbracket.$$

- If  $M$  deterministically converges to  $E[D(\vec{c})]$ , then  $\llbracket M \rrbracket_{\mathbb{S}}^0 = \llbracket E[D(\vec{c})] \rrbracket_{\mathbb{S}}^0 = \mathbf{0} \leq \llbracket M \rrbracket$ .
- About the inductive case, let us again distinguish three cases depending on the three cases of Lemma 47, applied to  $M$ :
  - If  $M$  deterministically diverges, then by Lemma 44,  $\llbracket M \rrbracket_{\mathbb{S}}^{n+1} \leq \llbracket M \rrbracket_{\mathbb{S}} = \llbracket M \rrbracket$ .
  - If  $M$  deterministically converges to a generalized value  $G$ , then by Lemma 45 and Lemma 46, it holds that

$$\llbracket M \rrbracket_{\mathbb{S}}^{n+1} = \llbracket G \rrbracket_{\mathbb{S}}^{n+1} \leq \llbracket G \rrbracket_{\mathbb{S}} = \delta(G) = \llbracket G \rrbracket = \llbracket M \rrbracket.$$

- If  $M$  deterministically converges to  $E[D(\vec{c})]$ , by Lemma 45 and Lemma 42 we have, by induction hypothesis, that

$$\begin{aligned} \llbracket M \rrbracket_{\mathbb{S}}^{n+1}(A) &= \llbracket E[D(\vec{c})] \rrbracket_{\mathbb{S}}^{n+1}(A) \\ &= \int \llbracket N \rrbracket_{\mathbb{S}}^n(A) E\{\mu_D(\vec{c})\}(dN) \\ &\leq \int \llbracket N \rrbracket(A) E\{\mu_D(\vec{c})\}(dN) \\ &= \llbracket M \rrbracket. \end{aligned}$$

□

RESTATEMENT OF THEOREM 3  $\llbracket M \rrbracket_{\mathbb{S}} = \llbracket M \rrbracket$ .

PROOF.

$$\begin{aligned} \llbracket M \rrbracket_{\Rightarrow} &= \sup_{n \in \mathbb{N}} \{ \mathcal{D} \mid M \rightarrow_n \mathcal{D} \} && \text{(by definition)} \\ &\leq \llbracket M \rrbracket_{\mathbb{S}} && \text{(by Lemma 48)} \\ &= \sup_{n \in \mathbb{N}} \llbracket M \rrbracket_{\mathbb{S}}^n && \text{(by Lemma 40)} \\ &\leq \llbracket M \rrbracket && \text{(by Lemma 49)} \\ &= \llbracket M \rrbracket_{\Rightarrow} && \text{(by Theorem 2)} \end{aligned}$$

□

**Corollary 1** *The measure  $\mathbf{P}_M$  is a sub-probability measure.*

## 4.6 Motivation for Bounded Scores

Recall that we only consider  $\text{score}(c)$  for  $c \in (0, 1]$ . Admitting  $\text{score}(2)$  (say), we exhibit an anomaly by constructing a recursive program that intuitively terminates with probability 1, but where the expected value of its score is infinite. Let  $\text{inflate} := \text{fix } f \lambda x. M_{\text{score}}$  where

$$M_{\text{score}} := \text{if } \text{flip}(0.5) \text{ then } \text{score}(2); (f \ x) \text{ else } x.$$

Let  $N = \text{inflate } V$ ; since  $\llbracket \text{score}(2) \rrbracket = \llbracket \text{fail} \rrbracket$  we have  $\llbracket N \rrbracket = 0.5 \cdot \delta(V)$ . However, evaluating  $N$  in a version of our trace semantics where the argument to  $\text{score}$  may be greater than 1 yields

$$\llbracket N \rrbracket_{\downarrow}^{\mathbb{S}}(\mathbb{S}_n) = \sum_{k=1}^n 1/2 = n/2$$

and so  $\llbracket N \rrbracket_{\mathbb{S}}(A) = \infty$  if  $V \in A$ , otherwise 0.

More strikingly, in the modified semantics we would also have  $\llbracket \text{inflate } \text{Gaussian}(0, 1) \rrbracket_{\mathbb{S}}([q, r]) = \infty$  for all real numbers  $q < r$ . This shows that general scoring in combination with recursion may yield measures that are not even  $\sigma$ -finite, causing many standard results in measure theory not to hold. For this reason, we restrict attention to positive scores bounded by one.

## 4.7 An Application of the Distribution-Based Semantics

It is routine to show that for all values  $V$ , it holds that  $\llbracket \text{score}(V) \rrbracket = \llbracket MV \rrbracket$ , where  $M$  is the term

$$\begin{aligned} &\lambda x. \text{if } (0 < x) \wedge (x \leq 1) \\ &\quad \text{then } (\text{if } \text{flip}(x) \text{ then true else } \Omega) \\ &\quad \text{else fail} \end{aligned}$$

This shows that even though  $\text{score}(V)$  and  $MV$  *do not* have the same sampling-based semantics, they can be used interchangeably whenever only their extensional, distribution-based behaviour is important. We use the equation to our advantage by encoding soft conditioning with score instead of flip (as discussed in Section 2.6), as the fewer the nuisance parameters the better for inference.

## 5 Inference

In this section, we are interested in sampling the return values of a particular closed term  $M \in C\Lambda$ . To avoid trivial cases, we assume that  $M$  has positive success probability and does not behave deterministically, i.e., that  $\llbracket M \rrbracket(\mathcal{V}) > 0$  and  $\llbracket M \rrbracket_{\downarrow}^{\mathbb{S}}(\{\emptyset\}) = 0$ . Our target distribution is the semantics of  $M$  conditioned on successful termination, that is,  $\llbracket M \rrbracket_{\mathcal{V}}(A) = \llbracket M \rrbracket(A \cap \mathcal{V}) / \llbracket M \rrbracket(\mathcal{V})$ . We can sample from the target distribution  $\llbracket M \rrbracket_{\mathcal{V}}$  using the Metropolis-Hastings (MH) algorithm [20, 12] over the space of traces  $s \in \mathbb{S}$ . This algorithm yields consecutive samples from a Markov chain over  $\mathbb{S}$ , such that the density of the samples  $s$  converges to  $\mathbf{P}_M^{\mathcal{V}}(s) / \llbracket M \rrbracket(\mathcal{V})$ . We can then apply the function  $\mathbf{O}_M$  to obtain the return value of  $M$  for a given trace. The algorithm is parametric in a proposal density function  $q(s, t)$  and consists of three steps:

1. Pick an initial state  $s$  with  $\mathbf{P}_M^{\mathcal{V}}(s) \neq 0$  (e.g., by running  $M$ ).
2. Draw the next state  $t$  at random with probability density  $q(s, t)$ .
3. Compute  $\alpha$  as below.

$$\alpha = \min \left( 1, \frac{\mathbf{P}_M^{\mathcal{V}}(t)}{\mathbf{P}_M^{\mathcal{V}}(s)} \cdot \frac{q(t, s)}{q(s, t)} \right)$$

- (a) With probability  $\alpha$ , output  $t$  and repeat from 2 with  $s := t$ .
- (b) Otherwise, output  $s$  and repeat from 2 with  $s$  unchanged.

The formula used for the number  $\alpha$  above is often called the Hastings *acceptance probability*. Different probabilistic programming language implementations use different choices for the density  $q$  above, based on pragmatics. The trivial choice would be to let  $q(s, t) = \mathbf{P}_M^\vee(t)$  for all  $s$ , which always yields  $\alpha = 1$  and so is equivalent to rejection sampling. We here define another simple density function  $q$  (based on Hur et al. [14]), giving emphasis to the conditions that it needs to satisfy in order to prove the convergence of the Markov chain given by the Metropolis-Hastings algorithm (Theorem 4).

## 5.1 A Metropolis-Hastings Transition Kernel

In the following, let  $M$  be a fixed program. Given a trace  $s = [c_1, \dots, c_n]$ , we write  $s_{i..j}$  for the trace  $[c_i, \dots, c_j]$  when  $1 \leq i \leq j \leq n$ . Intuitively, the following procedure describes how to obtain the proposal kernel density ( $q$  above):

1. Given a trace  $s$  of length  $n$ , let  $t = [t_1, \dots, t_n]$  where each  $t_i$  is drawn independently from a normal distribution with mean  $s_i$  and variance  $\sigma^2$ , and let  $p_i$  be the probability density of  $t_i$ .
2. Let  $k \leq n$  be the largest number such that  $(M, 1, []) \Rightarrow (M', w, t_{1..k})$ . There are three cases:
  - (a) If  $k = n$ , run  $M' \Downarrow_{t'}^{w'} V$ , and let  $q(s, t @ t') = p_1 \dots p_n w'$ .
  - (b) If  $k < n$  and  $M' \Downarrow_{[]}^1 V$ , let  $q(s, t_{1..k}) = p_1 \dots p_k$ .
  - (c) Otherwise, let  $q(s, t_{1..k}) = 0$  and propose the trace  $[]$ .

To define this kernel formally, we first give a function that partially evaluates  $M$  given a trace. Because of the possibility of nontermination, it is convenient to define such a function as a fixpoint of a higher-order function, since this simplifies the proof of measurability.

The set of closed terms  $C\Lambda$  is a CPO with respect to the partial order defined by  $\text{fail} \leq G$  for all  $G$ . Hence the set  $\mathbf{F}$  of all functions  $(C\Lambda \times \mathbb{S}) \rightarrow C\Lambda$  is a CPO with respect to the pointwise order. Define  $\Phi : \mathbf{F} \rightarrow \mathbf{F}$  as:

$$\Phi(f)(M, s) = \begin{cases} M & \text{if } s = [] \\ f(M', s') & \text{if } s \neq [], (M, 1, s) \rightarrow (M', w, s') \\ \text{fail} & \text{otherwise} \end{cases}$$

It is easy to check that  $\Phi$  is monotone and preserves suprema of  $\omega$  chains, so it is continuous. Hence, we can define:

$$\text{peval} = \sup_k \Phi^k(\lambda(M, s). \text{fail})$$

**Lemma 50** *For every  $M \in C\Lambda, c \in \mathbb{R}, s \in \mathbb{S}$ ,  $\text{peval}(\text{peval}(M, [c]), s) = \text{peval}(M, c :: s)$ .*

PROOF. By splitting the equality into two inequalities and using Scott induction.  $\square$

**Lemma 51** *For every  $M \in C\Lambda, c \in \mathbb{R}, s \in \mathbb{S}$ ,  $\text{peval}(\text{peval}(M, s), t) = \text{peval}(M, s @ t)$ .*

PROOF. By induction on  $|s|$ , with appeal to Lemma 50.  $\square$

For technical reasons, we need to ensure that  $Q$  is a probability kernel. We normalize  $q(s, \cdot)$  by giving non-zero probability  $q(s, [])$  to transitions ending in  $[]$  (which is not a completed trace of  $M$  by assumption).

**Transition Density  $q(s, t)$  and Kernel  $Q(s, A)$  for Program  $M$ :**

---


$$q(s, t) = (\prod_{i=1}^k \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)) \cdot \mathbf{P}_N^\vee(t_{k+1..|t|}) \text{ if } |t| \neq 0$$

where  $k = \min\{|s|, |t|\}$  and  $N = \text{peval}(M, t_{1..k})$



$$q(s, \square) = 1 - \int_A q(s, t) dt \text{ where } A = \{t \mid |t| \neq 0\}$$

$$Q(s, A) = \int_A q(s, t) dt$$


---

We prove the following lemmas in Appendix A.

**Lemma 52** *peval is a measurable function  $C\Lambda \times \mathbb{S} \rightarrow C\Lambda$ .*

**Lemma 53** *For any program  $M$ , the transition density  $q(\cdot, \cdot) : (\mathbb{S} \times \mathbb{S}) \rightarrow \mathbb{R}_+$  is measurable.*

**Lemma 54** *The function  $Q(s, A)$  is a probability kernel on  $(\mathbb{S}, \mathcal{S})$ .*

Hastings' Acceptance Probability  $\alpha$  is defined as

$$\alpha(s, t) = \min\left\{1, \frac{\mathbf{P}_M^\vee(t)q(t, s)}{\mathbf{P}_M^\vee(s)q(s, t)}\right\}$$

where we let  $\alpha(s, t) = 0$  if  $\mathbf{P}_M^\vee(t) = 0$  and otherwise  $\alpha(s, t) = 1$  if  $\mathbf{P}_M^\vee(s)q(s, t) = 0$ . Given the proposal transition kernel  $Q$  and the acceptance ratio  $\alpha$ , the Metropolis-Hastings algorithm yields a Markov chain over traces with the transition probability kernel.

$$P(s, A) = \int_A \alpha(s, t) Q(s, dt) + \delta(s)(A) \cdot \int (1 - \alpha(s, t)) Q(s, dt) \quad (1)$$

Define  $P^n(s, A)$  to be the probability of the  $n$ -th element of the chain with transition kernel  $P$  starting at  $s$  being in  $A$ :

$$\begin{aligned} P^0(s, A) &= \delta(s)(A) \\ P^{n+1}(s, A) &= \int P(t, A) P^n(s, dt) \end{aligned}$$

If  $f : X \rightarrow \mathbb{R}_+$  we let  $\text{supp}(f)$  be the *support* of  $f$ , that is,  $\{x \in X \mid f(x) \neq 0\}$ .

**Lemma 55** *If  $s_0 \in \text{supp}(\mathbf{P}_M^\vee)$  then  $P^n(s_0, \text{supp}(\mathbf{P}_M^\vee)) = 1$ .*

PROOF. By induction on  $n$ . The base case holds, since  $s_0 \in \text{supp}(\mathbf{P}_M^\vee)$  by assumption. For the induction case, we have  $P^{n+1}(s_0, \text{supp}(\mathbf{P}_M^\vee)) = \int P(s, \text{supp}(\mathbf{P}_M^\vee)) P^n(s_0, ds)$ . If  $s \in \text{supp}(\mathbf{P}_M^\vee)$  we have

$$\begin{aligned} P(s, \text{supp}(\mathbf{P}_M^\vee)) &= \int_{\text{supp}(\mathbf{P}_M^\vee)} \alpha(s, t) Q(s, dt) + \int (1 - \alpha(s, t)) Q(s, dt) \\ &= \int_{\text{supp}(\mathbf{P}_M^\vee)} q(s, t) dt + (1 - \alpha(s, \square))q(s, \square) \\ &= \int_{\text{supp}(\mathbf{P}_M^\vee)} q(s, t) dt + (1 - \alpha(s, \square))(1 - \int_{\text{supp}(\mathbf{P}_M^\vee)} q(s, t) dt) \\ &= 1 - \alpha(s, \square)(1 - \int_{\text{supp}(\mathbf{P}_M^\vee)} q(s, t) dt) \end{aligned}$$

where  $\alpha(s, \square) = 0$  since  $\mathbf{P}_M^\vee(\square) = 0$  by assumption. Then

$$\begin{aligned} \int P(s, \text{supp}(\mathbf{P}_M^\vee)) P^n(s_0, ds) &= \int_{\text{supp}(\mathbf{P}_M^\vee)} P(s, \text{supp}(\mathbf{P}_M^\vee)) P^n(s_0, ds) \\ &= \int_{\text{supp}(\mathbf{P}_M^\vee)} 1 P^n(s_0, ds) \\ &= 1 \end{aligned}$$

where the first and the third equality follow from the induction hypothesis.  $\square$

**Lemma 56** *There is  $0 \leq c < 1$  such that  $P^n(\square, \text{supp}(\mathbf{P}_M^\vee)) = 1 - c^n$  and  $P^n(\square, \{\square\}) = c^n$ .*

PROOF. Let  $c = 1 - \llbracket M \rrbracket_\downarrow^{\mathbb{S}\vee}(\mathbb{S} \setminus \{\square\})$ . By assumption  $\square \notin \text{supp}(\mathbf{P}_M^\vee)$  and  $c < 1$ , and since  $\llbracket M \rrbracket_\downarrow^{\mathbb{S}\vee}$  is a sub-probability distribution we have  $0 \leq c$ . We proceed by induction on  $n$ . The base case is trivial. For the induction case, we have  $P(s, \mathbb{S} \setminus \{\square\}) = 1$  for all  $s \in \text{supp}(\mathbf{P}_M^\vee)$ . Finally

$$P(\square, \text{supp}(\mathbf{P}_M^\vee)) = \int_{\text{supp}(\mathbf{P}_M^\vee)} \mathbf{P}_M^\vee = \llbracket M \rrbracket_\downarrow^{\mathbb{S}\vee}(\text{supp}(\mathbf{P}_M^\vee)) = \llbracket M \rrbracket_\downarrow^{\mathbb{S}\vee}(\mathbb{S} \setminus \{\square\}).$$

□

Based on Lemma 55 and 56, we below consider the Markov chain with kernel  $P$  restricted to  $\text{supp}(\mathbf{P}_M^\vee) \cup \{\square\}$ .

## 5.2 Correctness of Inference

By saying that the inference algorithm is correct, we mean that as the number of steps goes to infinity, the distribution of generated samples approaches the distribution specified by the sampling-based semantics of the program.

Formally, we define  $T^n(s, A) = P^n(s, \mathbf{O}_M^{-1}(A))$  as the value sample distribution at step  $n$  of the Metropolis-Hastings Markov chain. For two measures defined on the same measurable space  $(X, \Sigma)$ , we also define the variation norm  $\|\mu_1 - \mu_2\|$  as:

$$\|\mu_1 - \mu_2\| = \sup_{A \in \Sigma} |\mu_1(A) - \mu_2(A)|$$

We want to prove the following theorem:

**Theorem 4 (Correctness)** *For every trace  $s$  with  $\mathbf{P}_M^\vee(s) \neq 0$ ,*

$$\lim_{n \rightarrow \infty} \|T^n(s, \cdot) - \llbracket M \rrbracket_\vee\| = 0.$$

To do so, we first need to investigate convergence of  $P^n$ . It is convenient to define a measure for its target distribution.

### Target Distribution $\pi$

$$\pi(A) = \llbracket M \rrbracket_\downarrow^{\mathbb{S}\vee}(A) / \llbracket M \rrbracket_\downarrow^{\mathbb{S}\vee}(\mathbb{S})$$

We use a sequence of known results for Metropolis-Hastings Markov chains [30] to prove that  $P^n$  converges to  $\pi$ . We say that a Markov chain transition kernel  $P$  is  $\mathcal{D}$ -irreducible if  $\mathcal{D}$  is a non-zero sub-probability measure on  $(\mathbb{S}, \mathcal{S})$ , and for all  $x \in \mathbb{S}, A \in \mathcal{S}$  there exists an integer  $n > 0$  such that  $\mathcal{D}(A) > 0$  implies  $P^n(x, A) > 0$ . We say that  $P$  is  $\mathcal{D}$ -aperiodic if there do not exist  $d \geq 2$  and disjoint  $B_1, \dots, B_d$  such that  $\mathcal{D}(B_1) > 0$ , and  $x \in B_d$  implies  $P(x, B_1) = 1$ , and  $x \in B_i$  implies that  $P(x, B_{i+1}) = 1$  for  $i \in \{1, \dots, d-1\}$ .

**Lemma 57 (Tierney [30], Theorem 1 and Corollary 2)** *Let  $K$  be the transition kernel of a Markov chain given by the Metropolis-Hastings algorithm with target distribution  $\mathcal{D}$ . If  $K$  is  $\mathcal{D}$ -irreducible and aperiodic, then for all  $s$ ,  $\lim_{n \rightarrow \infty} \|K^n(s, \cdot) - \mathcal{D}\| = 0$ .*

**Lemma 58 (Strong Irreducibility)** *If  $\mathbf{P}_M^\vee(s) > 0$  and  $\llbracket M \rrbracket_\downarrow^{\mathbb{S}\vee}(A) > 0$  then  $P(s, A) > 0$ .*

PROOF. There is  $n$  such that  $\mathbf{P}_M^\vee(A \cap \mathbb{S}_n) > 0$ . Write  $A|_n = A \cap \mathbb{S}_n$ . For all  $t \in A|_n$ ,  $q(s, t) > 0$  by case analysis on whether  $n \leq |s|$ . If  $n \leq |s|$ , then for all  $t \in A|_n$ ,

$$\begin{aligned} q(s, t) &= \Pi_{i=1}^n \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) > 0 & \text{and} \\ q(t, s) &= (\Pi_{i=1}^n \text{pdf}_{\text{Gaussian}}(t_i, \sigma^2, s_i)) \cdot \mathbf{P}_{\text{peval}(M, s_{1..n})}^\vee(s_{(n+1)}, \dots, s_n) > 0. \end{aligned}$$

Similarly, if  $n > |s|$ , then for all  $t \in A|_n$ ,

$$\begin{aligned} q(s, t) &= (\prod_{i=1}^{|s|} \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)) \cdot \mathbf{P}_{\text{eval}(M, t_{1..|s|})}^{\mathcal{V}}(t_{(|s|+1)}, \dots, t_{|s|}) > 0 \quad \text{and} \\ q(t, s) &= \prod_{i=1}^{|s|} \text{pdf}_{\text{Gaussian}}(t_i, \sigma^2, s_i) > 0. \end{aligned}$$

Since  $\mu(A|_n) > 0$  and  $\mathbf{P}_M^{\mathcal{V}}(t) > 0$  for all  $t \in A|_n$ ,

$$\begin{aligned} P(s, A) &\geq P(s, A|_n) \\ &\geq \int_{A|_n} \alpha(s, t) Q(s, dt) \\ &= \int_{A|_n} \alpha(s, t) q(s, t) dt \\ &= \int_{A|_n} \min\{q(s, t), \frac{\mathbf{P}_M^{\mathcal{V}}(t) q(t, s)}{\mathbf{P}_M^{\mathcal{V}}(s)}\} dt \\ &> 0. \end{aligned}$$

□

**Corollary 2 (Irreducibility)** *P as given by Equation (1) is  $\pi$ -irreducible.*

**Lemma 59 (Aperiodicity)** *P as given by Equation (1) is  $\pi$ -aperiodic.*

PROOF. Assume that  $B_1, B_2$  are disjoint sets such that  $\pi(B_1) > 0$  and  $P(s, B_2) = 1$  for all  $s \in B_1$ . If  $s \in B_1$ , Lemma 58 gives that  $P(s, B_1) > 0$ , so  $P(s, B_2) < P(s, \mathbb{S}) = 1$ , which is a contradiction. A fortiori,  $P$  is  $\pi$ -aperiodic. □

**Lemma 60** *If  $\mu_1$  and  $\mu_2$  are measures on  $(X_1, \Sigma_1)$  and  $f : X_1 \rightarrow X_2$  is measurable  $\Sigma_1/\Sigma_2$ , then*

$$\|\mu_1 f^{-1} - \mu_2 f^{-1}\| \leq \|\mu_1 - \mu_2\|$$

PROOF. We have  $\sup_{B \in \Sigma_2} |\mu_1 f^{-1}(B) - \mu_2 f^{-1}(B)| = \sup_{A \in \Sigma'_1} |\mu_1(A) - \mu_2(A)|$ , where  $\Sigma'_1 = \{f^{-1}(B) | B \in \Sigma_2\}$ . By measurability of  $f$  we get  $\Sigma'_1 \subseteq \Sigma_1$ , so by monotonicity of sup we get  $\sup_{A \in \Sigma'_1} |\mu_1(A) - \mu_2(A)| \leq \sup_{A \in \Sigma_1} |\mu_1(A) - \mu_2(A)|$ . □

RESTATEMENT OF THEOREM 4 *For every trace  $s$  with  $\mathbf{P}_M^{\mathcal{V}}(s) \neq 0$ ,*

$$\lim_{n \rightarrow \infty} \|T^n(s, \cdot) - \llbracket M \rrbracket_{\mathcal{V}}\| = 0.$$

PROOF. By Corollary 2,  $P$  is  $\pi$ -irreducible, and by Lemma 59,  $P$  is  $\pi$ -aperiodic. Lemma 57 then yields that

$$\lim_{n \rightarrow \infty} \|P^n(x, \cdot) - \pi\| = 0.$$

By definition,  $T^n(s, A) = P^n(s, \mathbf{O}_M^{-1}(A))$  and  $\llbracket M \rrbracket_{\mathcal{V}}(A) = \llbracket M \rrbracket(A \cap \mathcal{V}) / \llbracket M \rrbracket(\mathcal{V})$ . By Theorem 3,  $\llbracket M \rrbracket(A \cap \mathcal{V}) = \llbracket M \rrbracket_{\mathbb{S}}(A \cap \mathcal{V}) = \llbracket M \rrbracket_{\downarrow}^{\mathbb{S}}(\mathbf{O}_M^{-1}(A \cap \mathcal{V})) = \llbracket M \rrbracket_{\downarrow}^{\mathbb{S}}(\mathbf{O}_M^{-1}(A) \cap \mathbf{O}_M^{-1}(\mathcal{V})) = \llbracket M \rrbracket_{\downarrow}^{\mathbb{S}\mathcal{V}}(\mathbf{O}_M^{-1}(A))$  and similarly  $\llbracket M \rrbracket(\mathcal{V}) = \llbracket M \rrbracket_{\downarrow}^{\mathbb{S}}(\mathbf{O}_M^{-1}(\mathcal{V})) = \llbracket M \rrbracket_{\downarrow}^{\mathbb{S}\mathcal{V}}(\mathbb{S})$ , which gives  $\llbracket M \rrbracket_{\mathcal{V}}(A) = \pi((\mathbf{O}_M^{-1}(A))$

Thus, by Lemma 60 and the squeeze theorem for limits we get

$$\lim_{n \rightarrow \infty} \|T^n(s, \cdot) - \llbracket M \rrbracket_{\mathcal{V}}\| \leq \lim_{n \rightarrow \infty} \|P^n(s, \cdot) - \pi\| = 0.$$

□

### 5.3 Examples

To illustrate how inference works, we revisit the geometric distribution and linear regression examples from Section 2. Before discussing the transition kernels for these models, note that the products of Gaussian densities always cancel out in the acceptance probability  $\alpha$ , because  $\text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) = \text{pdf}_{\text{Gaussian}}(t_i, \sigma^2, s_i)$  by the definition of the Gaussian PDF.

**Geometric distribution** Let us begin with the implementation of the geometric distribution, which we will call  $M_1$ . Since the only random primitive used in  $M_1$  is  $rnd$ , whose density is 1 on all its support, and there are no calls to **score**, the weight of any trace that yields a value must be 1. The set of valid traces is precisely  $S_1 = \{s \mid s_i \in [0, 0.5) \text{ for } i < |s| \wedge s_{|s|} \in [0.5, 1] \wedge |s| > 0\}$ . The proposal density is

$$q(s, t) = [t \in S_1] \Pi_{i=1}^k \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)$$

where  $k = \min\{|s|, |t|\}$ , so  $\alpha(s, t) = [t \in S_1]$ . This means that every valid trace is accepted. The transition kernel becomes

$$P(s, A) = \int_{A \cap S_1} \Pi_{i=1}^{\min\{|s|, |t|\}} \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) \mu(dt) \\ + [s \in A] \int_{S \setminus S_1} \Pi_{i=1}^{\min\{|s|, |t|\}} \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) \mu(dt)$$

**Linear regression with flip** In every trace in this model, which we call  $M_2$ , we have two draws from  $\text{Gaussian}(0, 2)$  followed by four observations encoded as calls to **flip**, which must all return **true** for a trace to be valid. We assume that **flip** is itself encoded using the uniform distribution, like in the previous example. Note that all valid traces have length 6. We have  $\mathbf{P}_{M_2}^\nu(s) = (\Pi_{i=1}^2 \text{pdf}_{\text{Gaussian}}(0, 2, s_i)) \left( \Pi_{i=1}^4 \left[ s_{i+2} \in \left[ 1, \frac{1}{e^{(s_i \cdot x_1 + s_2 - y_i)^2}} \right] \right] \right)$  when  $s \in \mathbb{R}^6$ . Note that the partial derivative of  $\mathbf{P}_{M_2}^\nu(s)$  with respect to each of  $s_3, s_4, s_5, s_6$  is zero wherever defined, precluding the use of efficient gradient-based methods for searching over these components of the trace.

Assuming  $\mathbf{P}_{M_2}^\nu(s) > 0$ , the proposal density is  $q(s, t) = (\Pi_{i=1}^6 \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)) \left( \Pi_{i=1}^4 [t_{i+2} \in [0, \frac{1}{e^{(t_i \cdot x_1 + t_2 - y_i)^2}}]] \right)$  for  $t \in \mathbb{R}^6$ . Hence, the acceptance ratio reduces to  $\alpha(s, t) = [t \in \mathbb{R}^6] \frac{\Pi_{i=1}^2 \text{pdf}_{\text{Gaussian}}(0, 2, t_i)}{\Pi_{i=1}^2 \text{pdf}_{\text{Gaussian}}(0, 2, s_i)} \Pi_{i=1}^4 [t_{i+2} \in [0, \frac{1}{e^{(t_i \cdot x_1 + t_2 - y_i)^2}}]]$ . Note that  $\alpha(s, t)$  only is positive if each of  $t_3, t_4, t_5, t_6$  are within a certain (small) interval. This is problematic for an implementation, since it will need to find suitable values for all these components of the trace for every new trace to be proposed, leading to inefficiencies due to a slowly mixing Markov chain.

**Linear regression with score** In this alternative version  $M_3$  of the previous model, we also have two draws from  $\text{Gaussian}(0, 2)$  at the beginning, but the calls to **flip** are replaced with calls to **score**, which modify the trace density without consuming any elements of the trace. Because the support of the Gaussian PDF is  $\mathbb{R}$ , the set of valid traces is precisely  $\mathbb{R}^2$ . We have  $\mathbf{P}_{M_1}^\nu(s) = [s \in \mathbb{R}^2] \Pi_{i=1}^2 \text{pdf}_{\text{Gaussian}}(0, 2, s_i)$ . Assuming  $\mathbf{P}_{M_2}^\nu(s) > 0$  again, we get the proposal density

$$q(s, t) = [t \in \mathbb{R}^2] \Pi_{i=1}^6 \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) \Pi_{i=1}^4 \frac{1}{e^{(t_i \cdot x_1 + t_2 - y_i)^2}}$$

where  $x = [0, 1, 2, 3]$  and  $y = [0, 1, 4, 6]$ . Thus, the acceptance ratio is

$$\alpha(s, t) = [t \in \mathbb{R}^2] \frac{\Pi_{i=1}^2 \text{pdf}_{\text{Gaussian}}(0, 2, t_i)}{\Pi_{i=1}^2 \text{pdf}_{\text{Gaussian}}(0, 2, s_i)} \frac{\Pi_{i=1}^4 e^{(t_i \cdot x_1 + t_2 - y_i)^2}}{\Pi_{i=1}^4 e^{(s_i \cdot x_1 + s_2 - y_i)^2}}$$

In contrast to the previous example, here the acceptance ratio is positive for all proposals, non-zero gradients exist almost everywhere, and there are four fewer nuisance parameters to deal with (one per data point!). This makes inference for this version of the model much more tractable in practice.

## 6 Related Work

To the best of our knowledge, the only previous theoretical justification for trace MCMC is the recent work by Hur et al. [14], who show correctness of trace MCMC for the imperative probabilistic

language R2 [21]. Their result does not apply to higher-order languages such as CHURCH or our  $\lambda$ -calculus. The authors do state that the space of traces in their language is equipped with a “stock” measure, and the probabilities of program traces and transitions can be treated as densities with respect to that measure. They do not, however, show that these densities are measurable. Their proof of correctness only shows that the acceptance ratio of the algorithm matches the standard formula for the MH algorithm on spaces of fixed dimensionality: the authors prove neither irreducibility nor aperiodicity of the resulting Markov chain.

Other probabilistic language implementations also use MCMC inference, including STAN [7], CHURCH [11], VENTURE [19], and ANGLICAN [31]. These works do not make formal correctness claims for their implementations, instead focusing on efficiency and convergence properties of their implementations.

Many recent probabilistic languages admit arbitrary positive scores. This is done either by having an explicit `score`-like function, as in STAN (called `increment_log_prob`) or WEBPPL (`factor`), or by observing that a particular value  $V$  was drawn from a given distribution  $D(\vec{c})$  (without adding it to the trace), as in WEB CHURCH (written  $(D \vec{c} V)$ ) or ANGLICAN (`observe`  $(D \vec{c} V)$ ). Recent work for a non-recursive  $\lambda$ -calculus by Staton et al. [27] also admits scores greater than one; the authors note that this introduces the possibility of “infinite model evidence errors”.

Wingate et al. [34] give a general program transformation for a probabilistic language to support trace MCMC, with a focus on labelling sample points in order to maximise sample reuse. Our trace semantics can be extended with such labelling, and we consider this important future work, given that Kiselyov [16] points out some difficulties with the algorithm, and proposes alternatives.

Kozen [17] gives a semantics of imperative probabilistic programs as partial measurable functions from infinite random traces to final states, and proves this semantics equivalent to a domain-theoretic one. Such operational semantics are akin to our `peval` function. Park et al. [23] study a similar semantics for functional programs, but “do not investigate measure-theoretic properties”. Cousot and Monerau [3] generalise Kozen’s operational semantics to consider probabilistic programs as measurable functions from a probability space into a semantics domain, and study abstract interpretation in this setting. Toronto et al. [33] use a pre-image version of Kozen’s semantics to obtain an efficient implementation using rejection sampling. Scibior et al. [26] define a monadic embedding of probabilistic programming in Haskell along the lines of Kozen’s semantics; their paper describes various inference algorithms but has no formal correctness results.

Like Kozen’s denotational semantics, our distributional semantics makes use of the partially additive structure on the category of sub-probability kernels [22] in order to treat programs that may take an unbounded number of computation steps.

While giving a fully abstract domain theory for probabilistic  $\lambda$ -calculi is known to be hard [15], there have been recent advances using probabilistic coherence spaces [4, 6] and game semantics [5], which in some cases are fully abstract. We see no strong obstacles in applying any of these to our calculus, but it is beyond the scope of this work. Another topic for future work are methodologies for equivalence checking in the style of logical relations or bisimilarity, which have been recently shown to work well in *discrete* probabilistic calculi [2].

## 7 Conclusions

As a foundation for probabilistic inference in languages such as CHURCH, we defined a probabilistic  $\lambda$ -calculus with draws from continuous probability distributions, defined its semantics as distributions on terms, and proved correctness of a trace MCMC inference algorithm via a sampling semantics for the calculus.

Although our emphasis has been on developing theoretical underpinnings, we also implemented our algorithm in F# to help develop our intuitions and indeed to help debug definitions. The algorithm is correct, effective, but not highly optimized. In future, we aim to extend our proofs to cover more efficient algorithms, inspired by Wingate et al. [34] and Kiselyov [16], for example.

## A Proofs of Measurability

This appendix contains the proofs of measurability of  $\mathbf{P}_M$ ,  $\mathbf{O}_M$ ,  $\mathbf{P}_M^\vee$ ,  $\mathbf{peval}$  and  $q$ , as well as a proof that  $Q$  is a probability kernel.

The proofs usually proceed by decomposing the functions into simpler operations. However, unlike Toronto [32], we do not define these functions entirely in terms of general measurable operators, because the scope for reuse is limited here. We would have, for instance, to define multiple functions projecting different subexpressions of different expressions, and prove them measurable. Hence, the overhead resulting from these extra definitions would be greater than the benefits.

First we recap some useful results from measure theory:

**Lemma 61 (Billingsley [1, ex. 13.1])** *Let  $(\Omega, \Sigma)$  and  $(\Omega', \Sigma')$  be two measurable spaces,  $T : \Omega \rightarrow \Omega'$  a function and  $A_1, A_2, \dots$  a countable collection of sets in  $\Sigma$  whose union is  $\Omega$ . Let  $\Sigma_n = \{A \mid A \subseteq A_n, A \in \Sigma\}$  be a  $\sigma$ -algebra in  $A_n$  and  $T_n : A_n \rightarrow \Omega'$  a restriction of  $T$  to  $A_n$ . Then  $T$  is measurable  $\Sigma/\Sigma'$  if and only if  $T_n$  is measurable  $\Sigma_n/\Sigma'$  for every  $n$ .*

- A function  $f : X_1 \rightarrow X_2$  between metric spaces  $(X_1, d_1)$  and  $(X_2, d_2)$  is *continuous* if for every  $x \in X_1$  and  $\epsilon > 0$ , there exists  $\delta$  such that for every  $y \in X_1$ , if  $d_1(x, y) < \delta$ , then  $d_2(f(x), f(y)) < \epsilon$ .
- A subset  $A$  of a metric space  $(X, d)$  is *dense* if

$$\forall x \in X, \epsilon > 0 \exists y \in A \quad d(x, y) < \epsilon$$

- A metric space is *separable* if it has a countable dense subset.
- Given a sequence of points  $x_n$  in a metric space  $(X, d)$ , we say that  $x$  is the *limit* of  $x_n$  if for all  $\epsilon > 0$ , there exists an  $N$  such that  $d(x_n, x) < \epsilon$ .
- A subset  $A$  of a metric space is *closed* if it contains all the limit points, that is if  $x_n \in A$  for all  $n$  and  $x_n \rightarrow x$ , then  $x \in A$ .

A convenient way of showing that a function is Borel-measurable is to show that it is continuous as a function between metric spaces.

Let us represent the product  $\sigma$ -algebra  $\mathcal{M} \times \mathcal{R} \times \mathcal{S}$  as a Borel  $\sigma$ -algebra induced by a metric. First, we define the standard metric on  $\mathbb{R}$ , and the disjoint union of Manhattan metrics for  $\mathbb{S}$ :

$$\begin{aligned} d_{\mathbb{R}}(w, w') &\triangleq |w - w'| \\ d_{\mathbb{S}}(s, s') &\triangleq \begin{cases} \sum_{i=1}^{|s|} |s_i - s'_i| & \text{if } |s| = |s'| \\ \infty & \text{otherwise} \end{cases} \end{aligned}$$

We can easily verify that  $(\mathbb{S}, d)$  generates  $\mathcal{S}$ . We define the metric on  $\Lambda \times \mathbb{R} \times \mathbb{S}$  to be the Manhattan metric:

$$d((M, w, s), (M', w', s')) \triangleq d_{\Lambda}(M, M') + d_{\mathbb{R}}(w, w') + d_{\mathbb{S}}(s, s')$$

The following is a standard result in measure theory:

**Lemma 62** *If  $X_1, X_2$  are separable metric spaces then*

$$\mathcal{B}(X_1 \times X_2) = \mathcal{B}(X_1) \times \mathcal{B}(X_2)$$

It is obvious that  $(\mathbb{R}, d)$  and  $(\mathbb{S}, d)$  are separable. Now, let  $\Lambda_Q$  be the subset of  $\Lambda_P$  in which all constants are rational. Then, it is easy to show that  $\Lambda_Q$  is countable.

**Lemma 63**  $\Lambda_Q$  is a dense subset of  $(\Lambda_P, d)$

PROOF. We need to prove that

$$\forall M \in \Lambda_P, \epsilon > 0 \exists M_Q \in \Lambda_Q \quad d(M, M_Q) < \epsilon$$

This can be easily shown by induction (the base case follows from the fact that  $\mathbb{Q}$  is a dense subset of  $\mathbb{R}$ ).  $\square$

**Lemma 64** The metric space  $(\Lambda_P, d)$  is separable.

PROOF. Corollary of Lemma 63.  $\square$

**Corollary 3** The  $\sigma$ -algebra on  $\Lambda \times \mathbb{R} \times \mathbb{S}$  generated by the metric  $d$  is  $\mathcal{M} \times \mathcal{R} \times \mathcal{S}$ .

Throughout this section, we call a function “measurable” if it is Borel measurable and “continuous” if it is continuous as a function between metric spaces.

We can use lemma 61 to split the space  $\mathcal{M}$  of expressions into subspaces of expressions of different type, and restrict functions (such as the reduction relation) to a given type of expression, to process different cases separately.

We write  $\mathbf{Subst}(M, x, v)$  for  $M\{V/x\}$ , to emphasize the fact that substitution is a function.

#### Detailed definition of substitution

$$\begin{aligned} \mathbf{Subst}(c, x, V) &\triangleq c \\ \mathbf{Subst}(x, x, V) &\triangleq V \\ \mathbf{Subst}(x, y, V) &\triangleq y \quad \text{if } x \neq y \\ \mathbf{Subst}(\lambda x.M, x, V) &\triangleq \lambda x.M \\ \mathbf{Subst}(\lambda x.M, y, V) &\triangleq \lambda x.(\mathbf{Subst}(M, y, V)) \quad \text{if } x \neq y \\ \mathbf{Subst}(M \ N, x, V) &\triangleq \mathbf{Subst}(M, x, V) \ \mathbf{Subst}(N, x, V) \\ \mathbf{Subst}(D(V_1, \dots, V_{|D|}), x, V) &\triangleq D(\mathbf{Subst}(V_1, x, V), \dots, \mathbf{Subst}(V_{|D|}, x, V)) \\ \mathbf{Subst}(g(V_1, \dots, V_{|g|}), x, V) &\triangleq g(\mathbf{Subst}(V_1, x, V), \dots, \mathbf{Subst}(V_{|g|}, x, V)) \\ \mathbf{Subst}(\text{if } W \text{ then } M \text{ else } L, x, V) &\triangleq \\ &\quad \text{if } \mathbf{Subst}(W, x, V) \text{ then } \mathbf{Subst}(M, x, V) \text{ else } \mathbf{Subst}(L, x, V) \\ \mathbf{Subst}(\text{score}(V'), x, V) &\triangleq \text{score}(\mathbf{Subst}(V', x, V)) \\ \mathbf{Subst}(\text{fail}, x, V) &\triangleq \text{fail} \end{aligned}$$

For convenience, let us also define a metric on contexts:

$$\begin{aligned} d([\cdot], [\cdot]) &\triangleq 0 \\ d(EM, FN) &\triangleq d(E, F) + d(M, N) \\ d((\lambda x.M)E, (\lambda x.N)F) &\triangleq d(M, N) + d(E, F) \\ d(E, F) &\triangleq \infty \quad \text{otherwise} \end{aligned}$$

**Lemma 65**  $d(E[M], F[N]) \leq d(E, F) + d(M, N)$ .

PROOF. By induction on the structure of  $E$ .

If  $d(E, F) = \infty$ , then the result is obvious, since  $d(M', N') \leq \infty$  for all  $M', N'$ .

Now let us assume  $d(E, F) \neq \infty$  and prove the result by simultaneous induction on the structure on  $E$  and  $F$ :

- Case  $E = F = [\cdot]$ : in this case,  $E[M] = M$ ,  $F[N] = N$ , and  $d(E, F) = 0$ , so obviously  $d(E[M], F[N]) = d(E, F) + d(M, N)$

- Case  $E = E' L_1, F = F' L_2$ :

We have  $d(E[M], F[N]) = d(E'[M] L_1, F'[N] L_2) = d(E'[M], F'[N]) + d(L_1, L_2)$ . By induction hypothesis,  $d(E'[M], F'[N]) \leq d(E', F') + d(M, N)$ , so  $d(E[M], F[N]) \leq d(E', F') + d(M, N) + d(L_1, L_2) = d(E, F) + d(M, N)$ .

- Case  $E = (\lambda x.L_1) E', F = (\lambda x.L_2) F'$ :

We have  $d(E[M], F[N]) = d((\lambda x.L_1)(E'[M]), (\lambda x.L_2)(F'[N])) = d(\lambda x.L_1, \lambda x.L_2) + d(E'[M], F'[N])$ . By induction hypothesis,  $d(E'[M], F'[N]) \leq d(E', F') + d(M, N)$ , so  $d(E[M], F[N]) \leq d(E', F') + d(\lambda x.L_1, \lambda x.L_2) + d(M, N) = d(E, F) + d(M, N)$ .  $\square$

**Lemma 66** *If  $d(E, F) = \infty$ , then for all  $R_1, R_2$ ,  $d(E[R_1], F[R_2])$ .*

PROOF. By induction on the structure of  $E$ :

- If  $E = []$ , then  $d(E, F) = \infty$  implies  $F \neq []$ :
  - If  $F = (\lambda x.M) F'$ , then  $d(E[R_1], F[R_2]) = d(R_1, (\lambda x.M) F'[R_2]) = \infty$ , because  $R_1$  is either not an application or of the form  $V_1 V_2$ , and  $F'[R_2]$  is not a value.
  - If  $F = F' N$ , then  $d(E[R_1], F[R_2]) = d(R_1, F'[R_2] N) = \infty$ , because  $R_1$  is either not an application or of the form  $V_1 V_2$ , and  $F'[R_2]$  is not a value.
- If  $E = (\lambda x.M) E'$ , then:
  - If  $F = F' N$ , then  $d(E[R_1], F[R_2]) = d(\lambda x.M, F'[R_2]) + d(E'[R_1], N) = \infty$ , because  $d(\lambda x.M, F'[R_2]) = \infty$ , as  $F'[R_2]$  cannot be a lambda-abstraction.
  - If  $F = (\lambda x.N) F'$ , then  $d(E, F) = \infty$  implies that either  $d(M, N) = \infty$  or  $d(E', F') = \infty$ . We have  $d(E[R_1], F[R_2]) = d(M, N) + d(E'[R_1], F'[R_2])$ . If  $d(M, N) = \infty$ , then obviously  $d(E[R_1], F[R_2]) = \infty$ . Otherwise, by induction hypothesis,  $d(E', F') = \infty$  gives  $d(E'[R_1], F'[R_2]) = \infty$ , and so  $d(E[R_1], F[R_2]) = \infty$ .
- If  $E = E' M$  and  $F = F' N$ , then  $d(E, F) = \infty$  implies that either  $d(M, N) = \infty$  or  $d(E', F') = \infty$ . We have  $d(E[R_1], F[R_2]) = d(M, N) + d(E'[R_1], F'[R_2])$ , so  $d(E'[R_1], F'[R_2]) = \infty$  follows like in the previous case.

The property also holds in all remaining cases by symmetry of  $d$ .  $\square$

**Lemma 67**  $d(E[R_1], F[R_2]) = d(E, F) + d(R_1, R_2)$ .

PROOF. If  $d(E, F) = \infty$ , then  $d(E[R_1], F[R_2]) = \infty$  by Lemma 66, otherwise the proof is the same as the proof of lemma 65, with inequality replaced by equality when applying the induction hypothesis.  $\square$

**Lemma 68**  $d(\text{Subst}(M, x, V), \text{Subst}(N, x, W)) \leq d(M, N) + k \cdot d(V, W)$  where  $k$  is the max of the multiplicities of  $x$  in  $M$  and  $N$

PROOF. By simultaneous induction on the structure of  $M$  and  $N$ .  $\square$

Let  $\mathcal{C}$  denote the set of contexts and  $\mathcal{G}$  the set of primitive functions. Let:

- $\Lambda_{appl} \triangleq \{E[(\lambda x.M)V] \mid E \in \mathcal{C}, M \in \Lambda, V \in \mathcal{V}\}$
- $\Lambda_{applt} \triangleq \{E[c V] \mid E \in \mathcal{C}, c \in \mathbb{R}, V \in \mathcal{V}\}$
- $\Lambda_{iftrue} \triangleq \{E[\text{if true then } M \text{ else } N] \mid E \in \mathcal{C}, M, N \in \Lambda\}$
- $\Lambda_{iffalse} \triangleq \{E[\text{if false then } M \text{ else } N] \mid E \in \mathcal{C}, M, N \in \Lambda\}$
- $\Lambda_{fail} \triangleq \{E[\text{fail}] \mid E \in \mathcal{C} \setminus \{[]\}\}$



- $\Lambda_{prim}(g) \triangleq \{E[g(\vec{c})] \mid E \in \mathcal{C}, \vec{c} \in \mathbb{R}^{|\mathcal{G}|}\}$
- $\Lambda_{prim} \triangleq \bigcup_{g \in \mathcal{G}} \Lambda_{prim}(g)$
- $A\Lambda_{if} \triangleq \{E[\text{if } G \text{ then } M \text{ else } N] \mid E \in \mathcal{C}, M, N \in \Lambda, G \in \mathcal{GV}\}$
- $\Lambda_{dist}(D) \triangleq \{E[D(\vec{c})] \mid E \in \mathcal{C}, \vec{c} \in \mathbb{R}^{|D|}\}$
- $\Lambda_{dist} \triangleq \bigcup_{D \in \mathcal{D}} \Lambda_{dist}(D)$
- $A\Lambda_{prim} \triangleq \bigcup_{g \in \mathcal{G}} E[g(G_1, \dots, G_{|g|})] \mid E \in \mathcal{C}, G_1, \dots, G_{|g|} \in \mathcal{GV}\}$
- $A\Lambda_{dist} \triangleq \bigcup_{D \in \mathcal{D}} E[D(G_1, \dots, G_{|D|})] \mid E \in \mathcal{C}, G_1, \dots, G_{|D|} \in \mathcal{GV}\}$
- $A\Lambda_{scr} \triangleq \{E[\text{score}(c)] \mid E \in \mathcal{C}, c \in \mathbb{R}\}$
- $\Lambda_{scr} \triangleq \{E[\text{score}(c)] \mid E \in \mathcal{C}, c \in (0, 1]\}$

**Lemma 69** *All the sets above are measurable.*

PROOF. All these sets except for  $\Lambda_{scr}$  are closed, so they are obviously measurable. The set  $\Lambda_{scr}$  is not closed (for example, we can define a sequence of points in  $\Lambda_{scr}$  whose limit is  $\text{score}(0) \notin \Lambda_{scr}$ ), but it is still measurable:

Define a function  $i_{scr} : A\Lambda_{scr} \rightarrow \mathbb{R}$  by  $i_{scr}(E[\text{score}(c)]) = c$ . This function is continuous and so measurable. Since the interval  $(0, 1]$  is a Borel subset of  $\mathbb{R}$ ,  $i_{scr}^{-1}((0, 1]) = \Lambda_{scr}$  is measurable.  $\square$

Now, we need to define the set of erroneous redexes of all types.

- $R\Lambda_{if} \triangleq A\Lambda_{if} \setminus (\Lambda_{iftrue} \cup \Lambda_{iffalse})$
- $R\Lambda_{prim} \triangleq A\Lambda_{prim} \setminus \Lambda_{prim}$
- $R\Lambda_{dist} \triangleq A\Lambda_{dist} \setminus \Lambda_{dist}$
- $R\Lambda_{scr} \triangleq A\Lambda_{scr} \setminus \Lambda_{scr}$
- $\Lambda_{error} \triangleq R\Lambda_{if} \cup R\Lambda_{prim} \cup R\Lambda_{dist} \cup R\Lambda_{scr}$

**Lemma 70** *The set  $\Lambda_{error}$  is measurable.*

PROOF. It is constructed from measurable sets by operations preserving measurability.  $\square$

Define:

$$\Lambda_{det} = \Lambda_{appl} \cup \Lambda_{cappl} \cup \Lambda_{iftrue} \cup \Lambda_{iffalse} \cup \Lambda_{fail} \cup \Lambda_{prim} \cup \Lambda_{error}$$

**Lemma 71**  *$\Lambda_{det}$  is measurable.*

PROOF.  $\Lambda_{det}$  is a union of measurable sets.  $\square$

**Lemma 72**  *$\mathcal{GV}$  is measurable.*

PROOF. It is easy to see that  $\mathcal{GV}$  is precisely the union of sets of all expressions of the form  $c$ ,  $\lambda x.M$ ,  $x$  and **fail**, so it is closed, and hence measurable.  $\square$

**Lemma 73**  *$\mathcal{V}$  is measurable.*

PROOF.  $\mathcal{V}$  is the union of sets of all expressions of the form  $c$ ,  $\lambda x.M$  and  $x$ , so it is closed, and hence measurable.  $\square$

## A.1 Deterministic reduction as a measurable function

Let us define a function performing one step of the reduction relation. This function has to be defined piecewise. Let us start with sub-functions reducing deterministic redexes of the given type.

$$\begin{aligned} g_{appl} & : \Lambda_{appl} \rightarrow \Lambda \\ g_{appl}(E[(\lambda x.M) V]) & = E[\mathbf{Subst}(M, x, v)] \end{aligned}$$

**Lemma 74**  *$g_{appl}$  is measurable.*

PROOF. By Lemma 67, we have  $d(E[(\lambda x.M)V], F[(\lambda x.N)W]) = d(E, F) + d(M, N) + d(V, W)$  and by Lemma 68,  $d(E[\mathbf{Subst}(M, x, V)], F[\mathbf{Subst}(N, x, W)]) \leq d(E, F) + d(M, N) + k \cdot d(V, W)$ , where  $k$  is the maximum of the multiplicities of  $x$  in  $M$  and  $N$ .

For any  $\epsilon > 0$ , take  $\delta = \frac{\epsilon}{k+1}$ . Then, if  $d(E[(\lambda x.M)V], F[(\lambda x.N)W]) < \delta$ , then

$$\begin{aligned} d(E[\mathbf{Subst}(M, x, V)], F[\mathbf{Subst}(N, x, W)]) & \leq d(E, F) + d(M, N) + k \cdot d(V, W) \\ & \leq (k+1) \cdot (d(E, F) + d(M, N) + d(V, W)) \\ & = (k+1) \cdot d(E[(\lambda x.M)V], F[(\lambda x.N)W]) \\ & < \epsilon \end{aligned}$$

Thus,  $g_{appl}$  is continuous, and so measurable.  $\square$

$$\begin{aligned} g_{applt} & : \Lambda_{applt} \rightarrow \Lambda \\ g_{applt}(E[c M]) & = E[\mathbf{fail}] \end{aligned}$$

**Lemma 75**  *$g_{applt}$  is measurable.*

PROOF. It is easy to check that  $g_{applt}$  is continuous.  $\square$

$$\begin{aligned} g_{prim} & : \Lambda_{prim} \rightarrow \Lambda \\ g_{prim}(E[g(\vec{c})]) & = E[\sigma_g(\vec{c})] \end{aligned}$$

**Lemma 76**  *$g_{prim}$  is measurable.*

PROOF. By assumption, every primitive function  $g$  is measurable.  $g_{prim}$  is a composition of a function splitting a context and a redex,  $g$  and a function combining a context with a redex, all of which are measurable.  $\square$

$$\begin{aligned} g_{iftrue} & : \Lambda_{iftrue} \rightarrow \Lambda \\ g_{iftrue}(E[\mathbf{if true then } M_1 \mathbf{ else } M_2]) & = E[M_1] \\ g_{iffalse} & : \Lambda_{iffalse} \rightarrow \Lambda \\ g_{iffalse}(E[\mathbf{if false then } M_1 \mathbf{ else } M_2]) & = E[M_2] \end{aligned}$$

**Lemma 77**  *$g_{iftrue}$  and  $g_{iffalse}$  are measurable.*

PROOF. We have  $d(E[\text{if true then } M_1 \text{ else } N_1], F[\text{if true then } M_2 \text{ else } N_2]) = d(E, F) + d(M_1, M_2) + d(N_1, N_2) \geq d(E[M_1], F[M_2])$ , so  $g_{iftrue}$  is continuous, and so measurable, and similarly for  $g_{iffalse}$ .  $\square$

$$\begin{aligned} g_{fail} &: \Lambda_{fail} \rightarrow \Lambda \\ g_{fail}(E[\text{fail}]) &= \text{fail} \end{aligned}$$

**Lemma 78**  $g_{fail}$  is measurable.

PROOF. Obvious, since it is a constant function.  $\square$

$$\begin{aligned} g_{error} &: \Lambda_{error} \rightarrow \Lambda \\ g_{error}(E[T]) &= E[\text{fail}] \end{aligned}$$

**Lemma 79**  $g_{error}$  is measurable.

PROOF. We have  $d(E[T_1], F[T_2]) \geq d(E, F) = d(E[\text{fail}], F[\text{fail}])$ , so  $g_{error}$  is continuous and hence measurable.  $\square$

$$\begin{aligned} g'_{det} &: \Lambda_{det} \rightarrow \Lambda \\ g'_{det} &= g_{appl} \cup g_{aplc} \cup g_{prim} \cup g_{iftrue} \cup g_{iffalse} \cup g_{fail} \cup g_{error} \end{aligned}$$

**Lemma 80**  $g'_{det}$  is measurable.

PROOF. Follows directly from Lemma 61.  $\square$

**Lemma 81**  $M \xrightarrow{det} N$  if and only if  $g'_{det}(M) = N$ .

PROOF. By inspection.  $\square$

## A.2 Small- step reduction as a measurable function

Let

$$\begin{aligned} \mathcal{T}_{val} &= \mathcal{GV} \times \mathbb{R} \times \mathbb{S} \\ \mathcal{T}_{det} &= \Lambda_{det} \times \mathbb{R} \times \mathbb{S} \\ \mathcal{T}_{scr} &= \Lambda_{scr} \times \mathbb{R} \times \mathbb{S} \\ \mathcal{T}_{rnd} &= \{(E[D(\vec{c})], w, c :: s) \mid E \in \mathcal{C}, D \in \mathcal{D}, \vec{c} \in \mathbb{R}^{|\mathcal{D}|}, w \in \mathbb{R}, s \in \mathbb{S}, c \in \mathbb{R}, \\ &\quad \text{pdf}_D(\vec{c}, c) > 0\} \end{aligned}$$

**Lemma 82**  $\mathcal{T}_{val}$ ,  $\mathcal{T}_{det}$ ,  $\mathcal{T}_{scr}$  and  $\mathcal{T}_{rnd}$  are measurable.

PROOF. The measurability of  $\mathcal{T}_{val}$ ,  $\mathcal{T}_{det}$  and  $\mathcal{T}_{scr}$  is obvious (they are products of measurable sets), so let us focus on  $\mathcal{T}_{rnd}$ .

For each distribution  $D$ , define a function  $i_D : \Lambda_{rnd}(D) \times \mathbb{R} \times (\mathbb{S} \setminus \{\emptyset\}) \rightarrow \mathbb{R}^{|\mathcal{D}|} \times \mathbb{R}$  by  $i_D(E[D(\vec{c})], w, c :: s) = (c, \vec{c})$ . This function is continuous, and so measurable. Then, since for each  $D$ ,  $\text{pdf}_D$  is measurable by assumption, the function  $j_d = \text{pdf}_D \circ i_D$  is measurable. Then,  $\mathcal{T}_{rnd} = \bigcup_{D \in \mathcal{D}} j_D^{-1}((0, \infty))$ , and since the set of distributions is countable,  $\mathcal{T}_{rnd}$  is measurable.  $\square$

Let  $\mathcal{T} = \Lambda \times \mathbb{R} \times \mathbb{S}$  and let  $\mathcal{T}_{blocked} = \mathcal{T} \setminus (\mathcal{T}_{val} \cup \mathcal{T}_{det} \cup \mathcal{T}_{scr} \cup \mathcal{T}_{rnd})$  be the set of non-reducible (“stuck”) triples, whose first components are not values. Obviously,  $\mathcal{T}_{blocked}$  is measurable.

Define:

$$\begin{aligned} g_{val} &: \mathcal{T}_{val} \rightarrow \mathcal{T} \\ g_{val}(G, w, s) &= (\mathbf{fail}, 0, []) \end{aligned}$$

Obviously,  $g_{val}$  is measurable.

$$\begin{aligned} g_{det} &: \mathcal{T}_{det} \rightarrow \mathcal{T} \\ g_{det}(M, w, s) &= (g'_{det}(M), w, s) \end{aligned}$$

**Lemma 83**  $g_{det}$  is measurable.

PROOF. All components of  $g_{det}$  are measurable. □

$$\begin{aligned} g_{rnd} &: \mathcal{T}_{rnd} \rightarrow \mathcal{T} \\ g_{rnd} &\triangleq (g_1, g_2, g_3) \\ g_1(E[\mathbf{D}(\vec{c})], w, c :: s) &\triangleq E[c] \\ g_2(E[\mathbf{D}(\vec{c})], w, c :: s) &\triangleq w \cdot \text{pdf}_{\mathbf{D}}(\vec{c}, c), \\ g_3(E[\mathbf{D}(\vec{c})], w, c :: s) &\triangleq s \end{aligned}$$

**Lemma 84**  $g_{rnd}$  is measurable.

PROOF. For  $g_1$ , we have  $d(E[c], E'[c']) \leq d(E, E') + d(c, c') \leq d(E, E') + d(\vec{c}, \vec{c}') + d(w, w') + d(s, s') = d((E[\mathbf{D}(\vec{c})], w, c :: s), (E'[\mathbf{D}(\vec{c}')] , w', c' :: s'))$  and  $d((E[\mathbf{D}(\vec{c})], w, c :: s), (E'[\mathbf{D}(\vec{c}')] , w', c' :: s')) = \infty$  if  $\mathbf{D} \neq \mathbf{E}$ , so  $g_1$  is continuous and hence Borel-measurable.

For  $g_2$ , we have  $g_2(E[\mathbf{D}(\vec{c})], w, c :: s) = g_w(E[\mathbf{D}(\vec{c})], w, c :: s) \times (\text{pdf}_{\mathbf{D}} \circ g_c)(E[\mathbf{D}(\vec{c})], w, c :: s)$ , where  $g_w(E[\mathbf{D}(\vec{c})], w, c :: s) = w$  and  $g_c(E[\mathbf{D}(\vec{c})], w, c :: s) = (\vec{c}, c)$ . The continuity (and so measurability) of  $g_w$  and  $g_c$  can be easily checked (as for  $g_1$  above). Thus,  $\text{pdf}_{\mathbf{D}} \circ g_c$  is a composition of measurable functions (since distributions are assumed to be measurable), and so  $g_2$  is a pointwise product of measurable real-valued functions, so it is measurable.

The continuity (and so measurability) of  $g_3$  can be shown in a similar way to  $g_1$ .

Hence, all the component functions of  $g_{rnd}$  are measurable, so  $g_{rnd}$  is itself measurable. □

$$\begin{aligned} g_{scr} &: \mathcal{T}_{scr} \rightarrow \mathcal{T} \\ g_{scr}(E[\mathbf{score}(c)], w, s) &\triangleq (E[\mathbf{true}], c \cdot w, s) \end{aligned}$$

**Lemma 85**  $g_{scr}$  is measurable.

PROOF. The first component function of  $g_{scr}$  can easily be shown continuous, and so measurable, and ditto for the third component. The second component is a pointwise product of two measurable functions, like in the  $g_{rnd}$  case. Hence,  $g_{scr}$  is measurable. □

For completeness, we also define:

$$\begin{aligned} g_{blocked} & : \mathcal{T}_{blocked} \rightarrow \mathcal{T} \\ g_{blocked}(M, w, s) & \triangleq (\mathbf{fail}, 0, []) \end{aligned}$$

This function is trivially measurable.  
Define

$$\begin{aligned} g & : \mathcal{T} \rightarrow \mathcal{T} \\ g & \triangleq g_{val} \cup g_{det} \cup g_{scr} \cup g_{blocked} \end{aligned}$$

**Lemma 86** *g is measurable.*

PROOF. Follows from Lemma 61. □

**Lemma 87** *For every  $(M, w, s) \in \mathcal{T}$ ,*

1. *If  $(M, w, s) \rightarrow (M', w', s')$ , then  $g(M, w, s) = (M', w', s')$ .*
2. *If  $g(M, w, s) = (M', w', s') \neq (\mathbf{fail}, 0, [])$ , then  $(M, w, s) \rightarrow (M', w', s')$ .*

PROOF. By inspection. □

### A.3 Measurability of $\mathbf{P}$ and $\mathbf{O}$

It is easy to check that the sets  $\Lambda$  and  $\mathbb{R}$  form cpos with the orderings  $\mathbf{fail} \leq M$  for all  $M$  and  $0 \leq x$ , respectively. This means that functions into  $\Lambda$  and  $\mathbb{R}$  also form cpos with pointwise ordering.

Define:

$$\begin{aligned} \Theta_{\Lambda}(f)(M, w, s) & \triangleq \begin{cases} M & \text{if } M \in \mathcal{GV}, s = [] \\ f(g(M, w, s)) & \text{otherwise} \end{cases} \\ \Theta_w(f)(M, w, s) & \triangleq \begin{cases} w & \text{if } M \in \mathcal{GV}, s = [] \\ f(g(M, w, s)) & \text{otherwise} \end{cases} \end{aligned}$$

It can be shown that these functions are continuous, so we can define:

$$\perp_{\Lambda} = (M, w, s) \mapsto \mathbf{fail}$$

$$\perp_w = (M, w, s) \mapsto 0$$

$$\mathbf{O}'(M, s) \triangleq \sup_n \Theta_{\Lambda}^n(\perp_{\Lambda})(M, 1, s)$$

$$\mathbf{P}'(M, s) \triangleq \sup_n \Theta_w^n(\perp_w)(M, 1, s)$$

**Lemma 88** *If  $(M, w_0, s) \Rightarrow (G, w, [])$ , then  $\sup_n \Theta_w^n(\perp_w)(M, w_0, s) = w$  and  $\sup_n \Theta_{\Lambda}^n(\perp_{\Lambda})(M, w_0, s) = G$ .*

PROOF. By induction on the derivation of  $(M, w_0, s) \Rightarrow (G, w, [])$ :

- If  $(M, w_0, s) \rightarrow^0 (G, w, [])$ , and so  $M \in \mathcal{GV}$  and  $s = []$ , then the equalities follow directly from the definitions of  $\Theta_w$  and  $\Theta_{\Lambda}$ .

- If  $(M, w_0, s) \rightarrow (M', w', s') \Rightarrow (G, w, \perp)$ , assume that  $\sup_n \Theta_w^n(\perp_w)(M', w', s') = w$  and  $\sup_n \Theta_\Lambda^n(\perp_\Lambda)(M', w', s') = G$ . We have  $M \notin \mathcal{GV}$ . By Lemma 87,  $g(M, w_0, s) = (M', w', s')$ . Hence  $\sup_n \Theta_w^n(\perp_w)(M, w_0, s) = \sup_n \Theta_w^n(\perp_w)(g(M, w_0, s)) = \sup_n \Theta_w^n(\perp_w)(M', w', s') = w$  by induction hypothesis. Similarly,  $\sup_n \Theta_\Lambda^n(\perp_\Lambda)(M, w_0, s) = G$ .  $\square$

**Corollary 4** *If  $(M, 1, s) \Rightarrow (G, w, \perp)$ , then  $\mathbf{P}'(M, s) = w$  and  $\mathbf{O}'(M, s) = G$ .*

**Lemma 89** *If  $\sup_n \Theta_w^n(\perp_w)(M, w_0, s) = w \neq 0$ , then  $(M, w_0, s) \Rightarrow (G, w, \perp)$  for some  $G \in \mathcal{GV}$ .*

PROOF. Because the supremum is taken with respect to a flat cpo,  $\sup_n \Theta_w^n(\perp_w)(M, w_0, s) = w > 0$  implies  $\Theta_w^k(\perp_w)(M, w_0, s) = w$  for some  $k > 0$ . We can then prove the result by induction on  $k$ :

- Base case,  $k = 1$ : We must have  $\Theta_w(\perp_w)(M, w_0, s) = w_0$ ,  $M = G \in \mathcal{GV}$  and  $s = \perp$  as otherwise we would obtain  $\perp_w(M, w_0, s) = 0$ . Hence  $(M, w_0, s)$  reduces to  $(G, w_0, \perp)$  in 0 steps.
- Induction step:  $\Theta_w^{k+1}(\perp_w)(M, w_0, s) = w$ . If  $M \in \mathcal{GV}$  and  $s = \perp$ , then  $w = w_0$  and  $(M, w_0, s)$  reduces to itself in 0 steps, like in the base case. Otherwise, we have  $\Theta_w^k(\perp_w)((M', w', s')) = w$ , where  $g(M, w_0, s) = (M', w', s')$ . We know that  $(M', w', s') \neq (\text{fail}, 0, \perp)$ , because otherwise we would have  $w = 0$ . Thus, by Lemma 87,  $(M, w_0, s) \rightarrow (M', w', s')$ . By induction hypothesis,  $(M', w', s') \Rightarrow (G, w, \perp)$ , which implies  $(M, w_0, s) \Rightarrow (G, w, \perp)$ .  $\square$

**Lemma 90** *If  $\sup_n \Theta_\Lambda^n(\perp_\Lambda)(M, w_0, s) = V \in \mathcal{V}$ , then  $(M, w_0, s) \Rightarrow (V, w, \perp)$  for some  $w \in \mathbb{R}$ .*

PROOF. Similar to the proof of Lemma 89.  $\square$

**Corollary 5** *If there are no  $G, w$  such that  $(M, 1, s) \Rightarrow (G, w, \perp)$ , then  $\mathbf{P}'(M, s) = 0$  and  $\mathbf{O}'(M, s) = \text{fail}$ .*

**Corollary 6** *For any  $M$ ,  $\mathbf{P}_M = \mathbf{P}'(M, \cdot)$  and  $\mathbf{O}_M = \mathbf{O}'(M, \cdot)$ .*

**Lemma 91** *If  $(X, \Sigma_1)$  and  $(Y, \Sigma_2)$  are measurable spaces,  $Y$  forms a flat cpo with a bottom element  $\perp$  such that  $\{\perp\} \in \Sigma_2$  and  $f_1, f_2, \dots$  is a  $\omega$ -chain of  $\Sigma_1/\Sigma_2$  measurable functions (on the cpo with pointwise ordering), then  $\sup_i f_i$  is  $\Sigma_1/\Sigma_2$  measurable.*

PROOF. Since  $f^{-1}(A \cup \{\perp\}) = f^{-1}(A) \cup f^{-1}(\{\perp\})$ , we only need to show that  $(\sup_i f_i)^{-1}(\{\perp\}) \in \Sigma_1$  and  $(\sup_i f_i)^{-1}(A) \in \Sigma_1$  for all  $A \in \Sigma_2$  such that  $\perp \notin A$ .

We have  $(\sup_i f_i)^{-1}(\{\perp\}) = \bigcap_i f_i^{-1}(\{\perp\})$ , which is measurable by definition. If  $\perp \notin A$ , then  $\sup_i f_i(x) \in A$  if and only if  $f_i(x) \in A$  for some  $i$ , so by extensionality of sets,  $\sup_i f_i^{-1}(A) = \bigcup_i f_i^{-1}(A) \subseteq \Sigma_1$ .  $\square$

**Lemma 92**  *$\mathbf{P}'$  is measurable.*

PROOF. First, let us show by induction on  $n$  that  $\Theta_w^n(\perp_w)$  is measurable for every  $n$ :

- Base case,  $n = 0$ :  $\Theta_w^0(\perp_w) = \perp_w$  is a constant function, and so trivially measurable.
- Induction step: suppose  $\Theta_w^n(\perp_w)$  is measurable. Then we have  $\Theta_w^{n+1}(\perp_w) = \Theta_w(\Theta_w^n(\perp_w))$ , so it is enough to show that  $\Theta_w(f)$  is measurable if  $f$  is measurable:

The domain of the first case is  $\mathcal{GV} \times \mathbb{R} \times \{\perp\}$ , which is clearly measurable. The domain of the second case is measurable as the complement of the above set in  $\mathcal{T}$ .

The sub-function corresponding to the first case returns the second component of its argument, so it is continuous and hence measurable. The second case is a composition of two measurable functions, hence measurable.

Thus,  $\Theta_w(f)$  is measurable for any measurable  $f$ , and so  $\Theta_w^{n+1}(\perp_w)$  is measurable.

By Lemma 91,  $\sup_n \Theta_w^n(\perp_w)$  is measurable. Since  $\mathbf{P}'$  is a composition of  $\sup_n \Theta_w^n(\perp_w)$  and a continuous function mapping  $(M, s)$  to  $(M, 1, s)$ , it is a composition of measurable functions, and so it is measurable.  $\square$

**Lemma 93**  $\mathbf{O}'$  is measurable.

PROOF. Similar to the proof of Lemma 92.  $\square$

RESTATEMENT OF LEMMA 27 For any closed term  $M$ , the function  $\mathbf{P}_M$  is measurable  $\mathbb{S} \rightarrow \mathbb{R}_+$ .

PROOF. Since  $\mathbf{P}'$  is measurable,  $\mathbf{P}_M = \mathbf{P}'(M, \cdot)$  is measurable for every  $M \in \Lambda$ .  $\square$

RESTATEMENT OF LEMMA 29 For each  $M$ , the function  $\mathbf{O}_M$  is measurable  $\mathbb{S} \rightarrow \mathcal{GV}$ .

PROOF. Since  $\mathbf{O}'$  is measurable,  $\mathbf{O}_M = \mathbf{O}'(M, \cdot)$  is measurable for every  $M \in \Lambda$ .  $\square$

**Lemma 94** For all  $M, s$ ,  $\mathbf{P}_M^\mathcal{V}(s) = \mathbf{P}_M(s)[\mathbf{O}_M(s) \in \mathcal{V}]$

PROOF. By Lemma 23, if  $M \Downarrow_s^w G$ , then  $w, G$  are unique. If  $M \Downarrow_s^w V$ , then  $\mathbf{P}_M(s) = w$ ,  $\mathbf{P}_M^\mathcal{V}(s)$  and  $\mathbf{O}_M(s) \in \mathcal{V}$ , so the equality holds. If  $M \Downarrow_s^w \text{fail}$ , then  $\mathbf{P}_M(s) = w$ ,  $\mathbf{P}_M^\mathcal{V}(s) = 0$  and  $\mathbf{O}_M(s) \notin \mathcal{V}$ , so both sides of the equation are 0. If there is no  $G$  such that  $M \Downarrow_s^w G$ , then both sides are also 0.  $\square$

RESTATEMENT OF LEMMA 30  $\mathbf{P}_M^\mathcal{V}$  is measurable for every  $M$ .

PROOF. By Lemma 94,  $\mathbf{P}_M^\mathcal{V}(s) = \mathbf{P}_M(s)[\mathbf{O}_M(s) \in \mathcal{V}]$ , so  $\mathbf{P}_M^\mathcal{V}$  is a pointwise product of a measurable function and a composition of  $\mathbf{O}_M$  and an indicator function for a measurable set, hence it is measurable.  $\square$

## A.4 Measurability of peval

For brevity, we write  $\lambda(M, s)$ .  $\text{fail}$  as  $\perp^\lambda$  below.

Recall the definition of  $\text{peval}$ :

$$\Phi(f)(M, s) = \begin{cases} M & \text{if } s = [] \\ f(M', s') & \text{if } s \neq [], (M, 1, s) \rightarrow (M', w, s') \\ \text{fail} & \text{otherwise} \end{cases}$$

$$\text{peval} = \sup_k \Phi^k(\perp^\lambda)$$

**Lemma 95** For every  $k$ ,  $\text{peval}_k = \Phi^k(\perp^\lambda)$  is measurable.

PROOF. By induction on  $k$ :

- Base case:  $k = 0$ :  $\text{peval}_0 = \perp^\lambda$  is a constant function on  $\Lambda \times \mathbb{S}$ , so trivially measurable.
- Induction step : we have  $\text{peval}_{k+1} = \Phi(\text{peval}_k)$ , so it is enough to show that  $\Phi(f)$  is measurable if  $f$  is measurable.  $\Phi(f)$  is defined in pieces, so we want to use Lemma 61.

The domain of the first case is  $\Lambda \times \{[]\}$ , so obviously measurable. The domain of the second case is  $p^{-1}(g^{-1}(\Lambda \times \mathbb{R} \times \mathbb{S}) \cap (\Lambda \times \{1\} \times (\mathbb{S} \setminus \{[]\})))$ , and  $p(M, s) = (M, 1, s)$  is continuous, and so measurable. Hence, the domain is measurable. Finally, the domain of the last case is the complement of the union of the two above measurable sets, which means it is also measurable.

Thus, we only need to show that the functions corresponding to these three cases are measurable. This is obvious in the first and third case, because the corresponding functions are constant. The function for the second case is  $\phi(M, s) = f(g(p(M, s)))$ , where  $p$  is as defined

above and  $g'$  is the restriction of  $g$  to  $g^{-1}(\Lambda \times \mathbb{R} \times \mathbb{S})$ , which is measurable since restrictions preserve measurability. Since composition of measurable functions is measurable,  $\phi$  is measurable.

Thus,  $\mathbf{peval}_{k+1}$  is measurable, as required.  $\square$

RESTATEMENT OF LEMMA 52  $\mathbf{peval}$  is a measurable function  $C\Lambda \times \mathbb{S} \rightarrow C\Lambda$ .

PROOF. Corollary of Lemmas 95 and 91.  $\square$

**Lemma 96** For every  $M \in C\Lambda, c \in \mathbb{R}, s \in \mathbb{S}$ ,  $\mathbf{peval}(\mathbf{peval}(M, [c]), s) \leq \mathbf{peval}(M, c :: s)$

PROOF. Define a property  $P \subseteq F$ :

$$P(f) \Leftrightarrow \sup_k \Phi^k(\perp^\lambda)(\Phi(f)(M, [c]), s) \leq \sup_k \Phi^k(\perp^\lambda)(M, c :: s) \quad \forall M, c, s$$

Since  $\sup_k \Phi^k(\perp^\lambda)$  is a fixpoint of  $\Phi$ ,  $\Phi(\sup_k \Phi^k(\perp^\lambda)) = \sup_k \Phi^k(\perp^\lambda)$ , so

$$P(\sup_k \Phi^k(\perp^\lambda)) \iff \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M, [c]), s) \leq \sup_k \Phi^k(\perp^\lambda)(M, c :: s) \quad \forall M, c, s$$

So we only need to prove  $P(\sup_k \Phi^k(\perp^\lambda))$ .

To show that the property  $P$  is  $\omega$ -inductive, let  $f_1 \leq f_2 \leq \dots$  be a  $\omega$ -chain and  $\sup_i f_i$  its limit. Then  $\Phi(f_1) \leq \Phi(f_2) \leq \dots$  and  $\sup_i \Phi(f_i) = \Phi(\sup_i f_i)$ . For all  $M, s$ , we have

$$\Phi(\sup_i f_i)(M, s) = (\sup_i \Phi(f_i))(M, s) = \sup_i (\Phi(f_i)(M, s))$$

Note that either  $\Phi(f_i)(M, s) = \mathbf{fail}$  for all  $i$  or there is some  $n$  such that  $\Phi(f_n)(M, s) \in \mathcal{V}$  and  $\Phi(f_m)(M, s) = \Phi(f_n)(M, s)$  for all  $m > n$ . In either case, there is a  $n(M, s)$  such that  $\sup_i \Phi(f_i)(M, s) = \Phi(f_{n(M, s)})(M, s)$ . Hence

$$\begin{aligned} P(\sup_i f_i) &\Leftrightarrow \sup_k \Phi^k(\perp^\lambda)(\Phi(\sup_i f_i)(M, [c]), s) \leq \sup_k \Phi^k(\perp^\lambda)(M, c :: s) \quad \forall M, c, s \\ &\Leftrightarrow \sup_k \Phi^k(\perp^\lambda)(\Phi(f_{n(M, [c])})(M, [c]), s) \leq \sup_k \Phi^k(\perp^\lambda)(M, c :: s) \quad \forall M, c, s \\ &\Leftrightarrow P(f_{n(M, [c])}) \end{aligned}$$

as required.

Now we can prove the desired property by Scott induction:

- Base case:

$$P(\perp^\lambda) \Leftrightarrow \sup_k \Phi^k(\perp^\lambda)(\Phi(\perp^\lambda)(M, [c]), s) \leq \sup_k \Phi^k(\perp^\lambda)(M, c :: s) \quad \forall M, c, s$$

For any  $M, c, s$ , we have

$$\sup_k \Phi^k(\perp^\lambda)(\Phi(\perp^\lambda)(M, [c]), s) = \sup_k \Phi^k(\perp^\lambda)(\mathbf{fail}, s) = \mathbf{fail} \leq \sup_k \Phi^k(\perp^\lambda)(M, c :: s)$$

as required.

- Induction step: We need to show that for all  $f$  such that  $P(f)$ ,  $P(\Phi(f))$  holds, that is

$$\sup_k \Phi^k(\perp^\lambda)(\Phi(\Phi(f))(M, [c]), s) \leq \sup_k \Phi^k(\perp^\lambda)(M, c :: s) \quad \forall M, c, s$$



– Case  $(M, 1, [c]) \rightarrow (M', w, [c])$ :

$$\begin{aligned}
LHS &= \sup_k \Phi^k(\perp^\lambda)(\Phi(\Phi(f))(M, [c]), s) \\
&= \sup_k \Phi^k(\perp^\lambda)(\Phi(f)(M', [c]), s) \\
(\text{by assumption}) &\leq \sup_k \Phi^k(\perp^\lambda)(M', c :: s) \\
&= \Phi(\sup_k \Phi^k(\perp^\lambda))(M, c :: s) \\
&= (\sup_k \Phi^k(\perp^\lambda))(M, c :: s) \\
&= RHS
\end{aligned}$$

– Case  $(M, 1, [c]) \rightarrow (M', w, [])$ :

$$\begin{aligned}
LHS &= \sup_k \Phi^k(\perp^\lambda)(\Phi(\Phi(f))(M, [c]), s) \\
&= \sup_k \Phi^k(\perp^\lambda)(\Phi(f)(M', []), s) \\
&= \sup_k \Phi^k(\perp^\lambda)(M', s) \\
&= \Phi(\sup_k \Phi^k(\perp^\lambda))(M, c :: s) \\
&= (\sup_k \Phi^k(\perp^\lambda))(M, c :: s) \\
&= RHS
\end{aligned}$$

– Case  $(M, 1, [c]) \not\rightarrow$ :

$$\begin{aligned}
LHS &= \sup_k \Phi^k(\perp^\lambda)(\Phi(\Phi(f))(M, [c]), s) \\
&= \sup_k \Phi^k(\perp^\lambda)(\text{fail}, s) \\
&= \Phi(\sup_k \Phi^k(\perp^\lambda))(M, c :: s) \\
&= (\sup_k \Phi^k(\perp^\lambda))(M, c :: s) \\
&= RHS
\end{aligned}$$

Therefore,  $P(\sup_k \Phi^k(\perp^\lambda))$ , holds, and so  $\text{peval}(\text{peval}(M, [c]), s) \leq \text{peval}(M, c :: s)$  for all closed  $M, c, s$ .  $\square$

**Lemma 97** *For every  $M \in C\Lambda, c \in \mathbb{R}, s \in \mathbb{S}$ ,  $\text{peval}(\text{peval}(M, [c]), s) \geq \text{peval}(M, c :: s)$*

PROOF. Like in the previous lemma, we use Scott induction. Define the property:

$$\begin{aligned}
Q(f) \quad \Leftrightarrow \quad & f(M, c :: s) \leq \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M, [c]), s) \quad \forall M, c, s \\
& \wedge f \leq \sup_k \Phi^k(\perp^\lambda)
\end{aligned}$$

We need to show that  $Q(\sup_k \Phi^k(\perp^\lambda))$  holds.

First, we need to verify that  $Q$  is  $\omega$ -inductive. This is obvious for the second conjunct, so let us concentrate on the first. Once again, we use the property that for all  $\omega$ -chains  $f_1 \leq f_2 \leq \dots$  and  $M, s$ , the chain  $f_1(M, s) \leq f_2(M, s) \leq \dots$  will eventually be stationary. For all  $M, c, s$ , we have  $(\sup_i f_i)(M, c :: s) = f_{n(M, c, s)}(M, c :: s)$  for some  $n(M, c, s)$ . Then for every  $M, c, s$ , the inequality

$$(\sup_i f_i)(M, c :: s) \leq \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M, [c]), s)$$

follows from  $Q(f_{n(M, c, s)})$ .

- Base case:

$$\begin{aligned} Q(\perp^\lambda) \quad \Leftrightarrow \quad \perp^\lambda(M, c :: s) &\leq \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M, [c]), s) \quad \forall M, c, s \\ &\wedge \perp^\lambda \leq \sup_k \Phi^k(\perp^\lambda) \end{aligned}$$

Both inequalities are obvious, because the LHS is always **fail**.

- Induction step: Give  $Q(f)$ , for every  $M, c, s$  we need to show:

$$\begin{aligned} \Phi(f)(M, c :: s) &\leq \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M, [c]), s) \quad \forall M, c, s \\ \Phi(f) &\leq \sup_k \Phi^k(\perp^\lambda) \end{aligned}$$

Again, the second inequality is obvious, so let us concentrate on the first.

- Case  $(M, 1, [c]) \rightarrow (M', w, [c])$ :

$$\begin{aligned} LHS &= f(M', c :: s) \\ \text{(by first assumption)} &\leq \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M', [c]), s) \\ &= \sup_k \Phi^k(\perp^\lambda)(\Phi(\sup_l \Phi^l(\perp^\lambda))(M, [c]), s) \\ &= \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M, [c]), s) \\ &= RHS \end{aligned}$$

- Case  $(M, 1, [c]) \rightarrow (M', w, [])$ :

$$\begin{aligned} LHS &= f(M', s) \\ \text{(by second assumption)} &\leq \sup_k \Phi^k(\perp^\lambda)(M', s) \\ &= \sup_k \Phi^k(\perp^\lambda)(\Phi(\sup_l \Phi^l(\perp^\lambda))(M', []), s) \\ &= \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M', []), s) \\ &= \sup_k \Phi^k(\perp^\lambda)(\Phi(\sup_l \Phi^l(\perp^\lambda))(M, [c]), s) \\ &= \sup_k \Phi^k(\perp^\lambda)(\sup_l \Phi^l(\perp^\lambda)(M, [c]), s) \\ &= RHS \end{aligned}$$

- Case  $(M, 1, [c]) \not\rightarrow$ :

$$\begin{aligned} LHS &= \mathbf{fail} \\ &\leq RHS \end{aligned}$$

As required. □

RESTATEMENT OF LEMMA 50  $\text{peval}(\text{peval}(M, [c]), s) = \text{peval}(M, c :: s)$

PROOF. Follows from Lemmas 96 and 97. □

RESTATEMENT OF LEMMA 51 *For all closed  $M, s, t$ ,  $\text{peval}(\text{peval}(M, s), t) = \text{peval}(M, s @ t)$*

PROOF. By induction on  $|s|$ .

- Base case:  $s = []$ . We have  $\text{peval}(M, []) = M$  (by definition of  $\text{peval}$ ), so the result is trivial.
- Induction step:  $s = c :: s'$ .

We want  $\text{peval}(\text{peval}(M, c :: s), t) = \text{peval}(M, c :: s @ t)$ .

We have:

$$\begin{aligned}
LHS &= \text{peval}(\text{peval}(M, c :: s), t) \\
&\text{(by Lemma 50)} = \text{peval}(\text{peval}(\text{peval}(M, [c]), s), t) \\
&\text{(by induction hypothesis)} = \text{peval}(\text{peval}(M, [c]), s @ t) \\
&\text{(by Lemma 50)} = \text{peval}(M, [c] :: s @ t)
\end{aligned}$$

□

## A.5 Measurability of $q$ and $Q$

**Lemma 98** For all  $s \in \mathbb{S}$ ,  $\int_{\mathbb{S} \setminus []} q(s, t) \mu(dt) \leq 1$

To prove this lemma, we need some auxiliary results:

**Lemma 99** If  $M \Downarrow_w^[] G$  and  $M \Downarrow_{w'}^s G'$ , then  $s = []$ .

PROOF. By induction on the derivation of  $M \Downarrow_w^[] G$ . □

**Lemma 100** If  $\mathbf{P}_M^\vee([]) > 0$ , then  $\mathbf{P}_M^\vee(t) = 0$  for all  $t \neq []$ .

PROOF. Follows directly from Lemma 99. □

**Lemma 101 (Tonelli's theorem for sums and integrals, 1.4.46 in [28])** If  $(\Omega, \Sigma, \mu)$  is a measure space and  $f_1, f_2, \dots$  a sequence of non-negative measurable functions, then

$$\int_{\Omega} \sum_{i=1}^{\infty} f_i(x) \mu(dx) = \sum_{i=1}^{\infty} \int_{\Omega} f_i(x) \mu(dx)$$

PROOF. Follows from the monotone convergence theorem. □

**Lemma 102 (Linearity of Lebesgue integral, 1.4.37 ii) from [28])** If  $(\Omega, \Sigma)$  is a measurable space,  $f$  a non-negative measurable function, and  $\mu_1, \mu_2, \dots$  a sequence of measures on  $\Sigma$ , then

$$\int_{\Omega} f(x) \sum_{i=1}^{\infty} \mu_i(dx) = \sum_{i=1}^{\infty} \int_{\Omega} f(x) \mu_i(dx)$$

**Lemma 103 (Ex. 1.4.36 xi) from [28])** If  $(\Omega, \Sigma, \mu)$  is a measure space and  $f$  a nonnegative measurable function on  $\Omega$  and  $B \in \Sigma$  and  $f^B$  a restriction of  $f$  to  $B$ , then

$$\int_{\Omega} f(x)[x \in B] \mu(dx) = \int_B f(x) \mu^B(dx)$$

Below we write  $q(s, t)$  as  $q_M(s, t)$ , to make the dependency on  $M$  explicit.

Let  $q_M^*$  be defined as follows:

$$q_M^*(s, t) = \begin{cases} \mathbf{P}_M^\vee([]) & \text{if } t = [] \\ q_M(s, t) & \text{otherwise} \end{cases}$$

**Lemma 104** For all  $M \in \Lambda$  and  $s, t \in \mathbb{S}$

$$q_M^*(s, t) = \begin{cases} \mathbf{P}_M^\mathcal{V}(t) & \text{if } s = [] \text{ or } t = [] \\ \text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1) q_{\text{peval}(M, [s_1])}^*([s_2, \dots, s_{|s|}], [t_2, \dots, t_{|t|}]) & \text{otherwise} \end{cases}$$

PROOF. By induction on  $|s|$ :

- Case  $s = []$ :

If  $t = []$ , the result follows directly from the definition of  $q_M^*$ . Otherwise,  $q_M^*([], t) = q_M([], t) = P_M^\mathcal{V}(t)$ , as required.

- Case  $|s| = n + 1 > 0$ :

Again, if  $t = []$ , the result follows immediately. Otherwise, we have

$$q_M^*(s, t) = q_M(s, t) = \Pi_{i=1}^k (\text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)) \mathbf{P}_{\text{peval}(M, [t_1, \dots, t_k])}^\mathcal{V}(t)$$

where  $k = \min(|s|, |t|) > 0$ . Hence

$$\begin{aligned} q_M^*(s, t) &= \text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1) \Pi_{i=2}^k (\text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)) \mathbf{P}_{\text{peval}(M, [t_1, \dots, t_k])}^\mathcal{V}([t_{k+1}, \dots, t_{|t|}]) \\ (\text{by Lemma 50}) &= \text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1) \Pi_{i=2}^k (\text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)) \mathbf{P}_{\text{peval}(\text{peval}(M, [t_1]), [t_2, \dots, t_k])}^\mathcal{V}([t_{k+1}, \dots, t_{|t|}]) \\ &= (\text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1)) q_{\text{peval}(M, [s_1])}^*([s_2, \dots, s_{|s|}], [t_2, \dots, t_{|t|}]) \end{aligned}$$

as required.  $\square$

**Lemma 105** If  $\mathbf{P}_M^\mathcal{V}([]) > 0$ , then  $\text{peval}(M, t) = \text{fail}$  for every  $t \neq []$ .

PROOF. It  $\mathbf{P}_M^\mathcal{V}([]) = w > 0$ , then we must have  $M \Downarrow_w^\mathcal{V} V$  for some  $V \in \mathcal{V}$ , which implies  $(M, 1, []) \Rightarrow (G, w, [])$ . Using Lemma 15, we can easily show by induction that  $(M, 1, t) \Rightarrow (G, w, t)$  for any  $t \neq []$ . Because the reduction relation is deterministic, this implies that there are no  $M'$ ,  $w'$  such that  $(M, 1, t) \Rightarrow (M', w', [])$  (if there were, we would have  $(M', w', []) \Rightarrow (G, w, t)$ , but no reduction rule can add an element to a trace). This means that  $\text{peval}$ , by applying reduction repeatedly, will never reach  $(M', [])$  for any  $M'$ , so  $\text{peval}(M, t) = \text{fail}$ .  $\square$

**Lemma 106** If  $\mathbf{P}_M^\mathcal{V}([]) > 0$ , then  $q_M^*(s, t) = 0$  for all  $s \in \mathbb{S}, t \neq []$ .

PROOF. Follows easily from Lemma 105.  $\square$

RESTATEMENT OF LEMMA 98 For all  $s \in \mathbb{S}$  and  $M \in \Lambda$ ,  $\int_{\mathbb{S} \setminus []} q_M(s, t) \mu(dt) \leq 1$

PROOF. By induction on  $|s|$ .

- Base case:  $s = []$

$$\begin{aligned} & \int_{\mathbb{S} \setminus \{[]\}} q_M([], t) \mu(dt) \\ &= \int_{\mathbb{S} \setminus \{[]\}} \mathbf{P}_M^\mathcal{V}(t) \mu(dt) \\ &\leq \int_{\mathbb{S}} \mathbf{P}_M(t) \mu(dt) \\ &= \llbracket M \rrbracket_{\downarrow}^\mathbb{S}(\mathbb{S}) \\ &= \llbracket M \rrbracket_{\downarrow}^\mathbb{S}(\mathbf{O}_M^{-1}(\mathcal{GV})) \\ &= \llbracket M \rrbracket_s(\mathcal{GV}) \\ \text{by Theorem 3} &= \llbracket M \rrbracket(\mathcal{GV}) \\ &\leq 1 \end{aligned}$$

because  $\llbracket M \rrbracket$  is a subprobability measure.

- Induction step:  $s \neq \emptyset$

We have:

$$\begin{aligned}
& \int_{\mathbb{S} \setminus \{\emptyset\}} q_M(s, t) \mu(dt) \\
&= \int_{\mathbb{S} \setminus \{\emptyset\}} q_M^*(s, t) \mu(dt) \\
\text{(by Thm 16.9 from Billingsley)} &= \sum_{i=1}^{\infty} \int_{\mathbb{R}^i} q_M^*(s, t) \mu(dt) \\
\text{(by Lemma 103)} &= \sum_{i=1}^{\infty} \int_{\mathbb{R}^i} q_M^*(s, t) \lambda^i(dt) \\
&= \sum_{i=1}^{\infty} \int_{\mathbb{R}^i} \text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1) q_{\text{peval}(M, [t_1])}^*([s_2, \dots, s_{|s|}], [t_2, \dots, t_{|t|}]) \lambda^i(dt) \\
\text{(by Fubini's theorem)} &= \sum_{i=1}^{\infty} \int_{\mathbb{R}} \text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1) \int_{\mathbb{R}^{i-1}} q_{\text{peval}(M, [t_1])}^*(s', t') \lambda^{i-1}(dt') \lambda(dt_1) \\
\text{(by Lemma 101)} &= \int_{\mathbb{R}} \text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1) \sum_{i=0}^{\infty} \int_{\mathbb{R}^i} q_{\text{peval}(M, [t_1])}^*(s', t') \lambda^i(dt') \lambda(dt_1) \\
&= \int_{\mathbb{R}} \text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1) \left( \int_{\{\emptyset\}} \mathbf{P}_{\text{peval}(M, [t_1])}^{\mathcal{V}}(t') \mu(dt') + \right. \\
&\quad \left. \int_{\mathbb{S} \setminus \{\emptyset\}} q_{\text{peval}(M, [t_1])}^*(s', t') \mu(dt') \right) \lambda(dt_1)
\end{aligned}$$

Now, we need to show that for all  $N$ ,

$$\int_{\{\emptyset\}} \mathbf{P}_N^{\mathcal{V}}(t') \mu(dt') + \int_{\mathbb{S} \setminus \{\emptyset\}} q_N^*(s', t') \mu(dt') \leq 1 \quad (2)$$

First, note that  $\int_{\{\emptyset\}} \mathbf{P}_N^{\mathcal{V}}(t') \mu(dt') \leq \int_{\mathbb{S}} \mathbf{P}_N^{\mathcal{V}}(t') \mu(dt') \leq 1$ , by the same property as the one used in the base case. We also have  $\int_{\{\emptyset\}} \mathbf{P}_N^{\mathcal{V}}(t') \mu(dt') = \mathbf{P}_N^{\mathcal{V}}(\emptyset)$ , so by Lemma 106, if  $\mathbf{P}_N^{\mathcal{V}}(\emptyset) > 0$ , then

$$\int_{\{\emptyset\}} \mathbf{P}_N^{\mathcal{V}}(t') \mu(dt') + \int_{\mathbb{S} \setminus \{\emptyset\}} q_N^*(s', t') \mu(dt') = \int_{\{\emptyset\}} \mathbf{P}_N^{\mathcal{V}}(t') \mu(dt') \leq 1$$

On the other hand, if  $\mathbf{P}_N^{\mathcal{V}}(\emptyset) = 0$ , then

$$\int_{\{\emptyset\}} \mathbf{P}_N^{\mathcal{V}}(t') \mu(dt') + \int_{\mathbb{S} \setminus \{\emptyset\}} q_N^*(s', t') \mu(dt') = \int_{\mathbb{S} \setminus \{\emptyset\}} q_N^*(s', t') \mu(dt') = \int_{\mathbb{S} \setminus \{\emptyset\}} q_N(s', t') \mu(dt') \leq 1$$

by induction hypothesis.

Hence:

$$\begin{aligned}
& \int_{\mathbb{S} \setminus \{\emptyset\}} q_M(s, t) \mu(dt) \\
&\leq \int_{\mathbb{R}} \text{pdf}_{\text{Gaussian}}(s_1, \sigma^2, t_1) \lambda(dt_1) \\
&= 1
\end{aligned}$$

as required. □

RESTATEMENT OF LEMMA 53 *For any program  $M$ , the transition density  $q(\cdot, \cdot) : (\mathbb{S} \times \mathbb{S}) \rightarrow \mathbb{R}_+$  is measurable.*

PROOF. It is enough to show that  $q(s, t)$  is measurable for every  $|s| = n$  and  $|t| = m$ , then the result follows from Lemma 61.

Note that a function taking a sequence  $s$  and returning any subsequence of it is trivially continuous and measurable, so for any function of  $s$  and  $t$  to be measurable, it is enough to show that it is measurable as a function of some projections of  $s$  and  $t$ .

- If  $m > 0$  and  $n < m$ , then we have  $q(s, t) = \prod_{i=1}^n \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) \mathbf{P}_{\text{peval}(M, t_{1..n})}^{\mathcal{V}}(t_{n+1..m}) = \prod_{i=1}^n \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) \mathbf{P}'(\text{peval}(M, t_{1..n}), t_{n+1..m}) [\mathbf{O}'(\text{peval}(M, t_{1..n}), t_{n+1..m}) \in \mathcal{V}]$ .

Each  $\text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)$  is measurable, as a composition of a function projecting  $(s_i, t_i)$  from  $(s, t)$  and the Gaussian pdf, so their pointwise product must be measurable.

Now,  $\mathbf{P}'$  is measurable, and the function mapping  $(s, t)$  to  $(\text{peval}(M, t_{1..n}), t_{n+1..m})$  is a pair of two measurable functions, one of which is a composition of the measurable  $\text{peval}(M, \cdot)$  and a projection of  $t_{1..n}$ , and the other just a projection of  $t_{n+1..m}$ . Hence, the function mapping  $(s, t)$  to  $\mathbf{P}'(\text{peval}(M, t_{1..n}), t_{n+1..m})$  is a composition of measurable functions.

Finally,  $[\mathbf{O}'(\text{peval}(M, t_{1..n}), t_{n+1..m}) \in \mathcal{V}]$  is a composition of the measurable function mapping  $(s, t)$  to  $(\text{peval}(M, t_{1..n}), t_{n+1..m})$  and the indicator function for the measurable set  $\mathcal{V}$ , thus it is measurable.

Hence,  $q(s, t)$  is a pointwise product of measurable functions, so it is measurable.

- If  $m > 0$  and  $n \geq m$ , then  $q(s, t) = \prod_{i=1}^m \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) \mathbf{P}_{\text{peval}(M, t)}^{\mathcal{V}}(\square) = \prod_{i=1}^m \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i) \mathbf{P}'(\text{peval}(M, t), \square) [\mathbf{O}'(\text{peval}(M, t), \square) \in \mathcal{V}]$ .

Now, the function mapping  $(s, t)$  to  $\prod_{i=1}^m \text{pdf}_{\text{Gaussian}}(s_i, \sigma^2, t_i)$  is measurable like in the previous case. The function mapping  $(s, t)$  to  $(\text{peval}(M, t), \square)$  is a pairing of two measurable functions, one being a composition of the projection of  $t$  and  $\text{peval}(M, \cdot)$ , the other being a constant function returning  $\square$ . Hence,  $\mathbf{P}'(\text{peval}(M, t), \square)$  is a composition of two measurable functions. Meanwhile,  $[\mathbf{O}'(\text{peval}(M, t), \square) \in \mathcal{V}]$  is a composition of a measurable function and an indicator function.

- If  $m = 0$ , then  $q(s, \square) = 1 - \int_{\mathbb{S} \setminus \{\square\}} q(s, t) \mu(dt)$ . Since we have already shown that  $q(s, t)$  is measurable on  $\mathbb{S} \times (\mathbb{S} \setminus \{\square\})$ ,  $\int_{\mathbb{S} \setminus \{\square\}} q(s, t) \mu(dt)$  is measurable by Fubini's theorem, so  $q(s, \square)$  is a difference of measurable functions, and hence it is measurable.  $\square$

RESTATEMENT OF LEMMA 54 *The function  $Q$  is a probability kernel on  $(\mathbb{S}, \mathcal{S})$ .*

PROOF. We need to verify the two properties of probability kernels:

1. For every  $s \in \mathbb{S}$ ,  $Q(s, \cdot)$  is a probability measure on  $\mathbb{S}$ . Since for every  $s \in \mathbb{S}$ ,  $q(s, \cdot)$  is non-negative measurable  $\mathcal{S}$  (by [1, Theorem 18.1]),  $Q(s, B) = \int_B q(s, y) \mu(dy)$  (as a function of  $B$ ) is a well-defined measure for all  $s \in \mathbb{S}$ . Finally,  $Q(s, \mathbb{S}) = Q(s, \square) + Q(s, \mathbb{S} \setminus \{\square\}) = 1$ .
2. For every  $B \in \mathcal{S}$ ,  $Q(\cdot, B)$  is a non-negative measurable function on  $\mathbb{S}$ : Since  $(\mathbb{S}, \mathcal{S}, \mu)$  is a  $\sigma$ -finite measure space,  $q(\cdot, \cdot)$  is non-negative and measurable  $\mathcal{S} \times \mathcal{S}$  and  $Q(s, B) = \int_B q(s, y) \mu(dy)$ , this follows from [1, Theorem 18.3].  $\square$

## References

- [1] P. Billingsley. *Probability and Measure*. Wiley-Interscience, third edition, 1995.
- [2] A. Bizjak and L. Birkedal. Step-indexed logical relations for probability. In A. M. Pitts, editor, *Proceedings of FoSSaCS 2015*, volume 9034 of *LNCS*, pages 279–294. Springer, 2015. doi: 10.1007/978-3-662-46678-0\_18.

- [3] P. Cousot and M. Monerau. Probabilistic abstract interpretation. In *Proceedings of ESOP 2012*, volume 7211 of *LNCS*, pages 166–190. Springer, 2012.
- [4] V. Danos and T. Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Information and Computation*, 209(6):966–991, 2011.
- [5] V. Danos and R. Harmer. Probabilistic game semantics. *ACM Transactions on Computational Logic*, 3(3):359–382, 2002.
- [6] T. Ehrhard, C. Tasson, and M. Pagani. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In *Proceedings of POPL 2014*, pages 309–320, 2014.
- [7] A. Gelman, D. Lee, and J. Guo. Stan: A probabilistic programming language for Bayesian inference and optimization. *Journal of Educational and Behavioral Statistics*, 40(5):530–543, 2015. doi: 10.3102/1076998615606113.
- [8] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
- [9] N. D. Goodman and A. Stuhlmüller. The design and implementation of probabilistic programming languages. <http://dippl.org>, 2014.
- [10] N. D. Goodman and J. B. Tenenbaum. Probabilistic models of cognition. <http://probmods.org>, 2014.
- [11] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: a language for generative models. In D. A. McAllester and P. Myllymäki, editors, *Proceedings of UAI 2008*, pages 220–229. AUAI Press, 2008.
- [12] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970. doi: 10.1093/biomet/57.1.97.
- [13] M. D. Homan and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, Jan. 2014. ISSN 1532-4435. URL <http://dl.acm.org/citation.cfm?id=2627435.2638586>.
- [14] C. Hur, A. V. Nori, S. K. Rajamani, and S. Samuel. A provably correct sampler for probabilistic programs. In P. Harsha and G. Ramalingam, editors, *Proceedings of FSTTCS 2015*, volume 45 of *LIPICs*, pages 475–488. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi: 10.4230/LIPICs.FSTTCS.2015.475.
- [15] C. Jones and G. D. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of LICS 1989*, pages 186–195, 1989.
- [16] O. Kiselyov. Problems of the lightweight implementation of probabilistic programming, Jan. 2016. Poster at PPS’2016 workshop.
- [17] D. Kozen. Semantics of probabilistic programs. In *Proceedings of FOCS 1979*, pages 101–114. IEEE Computer Society, 1979. doi: 10.1109/SFCS.1979.38.
- [18] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.
- [19] V. K. Mansinghka, D. Selsam, and Y. N. Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *CoRR*, abs/1404.0099, 2014. URL <http://arxiv.org/abs/1404.0099>.
- [20] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

- [21] A. V. Nori, C. Hur, S. K. Rajamani, and S. Samuel. R2: an efficient MCMC sampler for probabilistic programs. In C. E. Brodley and P. Stone, editors, *Proceedings of AAAI 2014.*, pages 2476–2482. AAAI Press, 2014.
- [22] P. Panangaden. The category of Markov kernels. *ENTCS*, 22:171–187, 1999. In proceedings of PROBMIV 1998.
- [23] S. Park, F. Pfenning, and S. Thrun. A probabilistic language based upon sampling functions. In J. Palsberg and M. Abadi, editors, *Proceedings of POPL 2005*, pages 171–182. ACM, 2005.
- [24] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.
- [25] S. J. Russell. Unifying logic and probability. *Commun. ACM*, 58(7):88–97, 2015. doi: 10.1145/2699411.
- [26] A. Scibior, Z. Ghahramani, and A. D. Gordon. Practical probabilistic programming with monads. In B. Lippmeier, editor, *Proceedings of Haskell 2015*, pages 165–176. ACM, 2015. doi: 10.1145/2804302.2804317.
- [27] S. Staton, H. Yang, C. Heunen, O. Kammar, and F. Wood. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. *CoRR*, abs/1601.04943, 2016. URL <http://arxiv.org/abs/1601.04943>.
- [28] T. Tao. *An Introduction to Measure Theory*. AMS, 2011.
- [29] S. Thrun. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.
- [30] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994. doi: 10.1214/aos/1176325750.
- [31] D. Tolpin, J. van de Meent, and F. Wood. Probabilistic programming in Anglican. In A. Bifet, M. May, B. Zadrozny, R. Gavaldà, D. Pedreschi, F. Bonchi, J. S. Cardoso, and M. Spiliopoulou, editors, *Proceedings of ECML PKDD 2015, Part III*, volume 9286 of *LNCS*, pages 308–311. Springer, 2015. doi: 10.1007/978-3-319-23461-8\_36.
- [32] N. Toronto. *Useful Languages for Probabilistic Modeling and Inference*. PhD thesis, Brigham Young University, Provo, UT, 2014. URL <https://www.cs.umd.edu/~ntoronto/papers/toronto-2014diss.pdf>.
- [33] N. Toronto, J. McCarthy, and D. V. Horn. Running probabilistic programs backwards. In J. Vitek, editor, *Proceedings of ESOP 2015*, volume 9032 of *LNCS*, pages 53–79. Springer, 2015. doi: 10.1007/978-3-662-46669-8\_3.
- [34] D. Wingate, A. Stuhlmüller, and N. D. Goodman. Lightweight implementations of probabilistic programming languages via transformational compilation. In *Proceedings of AISTATS 2011*, page 131, 2011.